Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему : «Розроблення програмного забезпечення для обліку робочого часу працівників на підприємстві»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Щербина Олександр Олександрович
(прізвище та ініціали)
Керівник д.е.н., проф. Корнєєв М.В.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Професор кафедри кібербезпеки та інформаційних технологій
(посада)
д.е.н., професор Паршина О.А.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Щербина О.О. Розроблення програмного забезпечення для обліку робочого часу працівників на підприємстві.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

У даній кваліфікаційній роботі розглянуто процес проектування та реалізації системи обліку робочого часу працівників.

Проаналізовано існуюче програмне забезпечення, що є популярним на міжнародному ринку та має велику кількість клієнтів у різних країнах.

Також у першій частині описано, що представляє собою аналіз предметної області.

У другому розділі був проведений аналіз інструментів для розроблення програмного забезпечення обліку робочого часу працівників. Були обрані сучасні та ефективні технології та компоненти.

Використовуючи мову програмування C# та базу даних MSSQLServer, розроблено програмне забезпечення, яке дозволяє ефективно відстежувати часові ресурси робітників та автоматизувати процеси обліку. Робота включає в себе аналіз потреб предметної області, проектування бази даних, розробку програмного забезпечення та його тестування. Ця система має важливе значення для підприємств, оскільки дозволяє оптимізувати використання робочого часу, покращує ефективність управління та допомагає у зборі та аналізі даних. Результати роботи можуть бути використані для підвищення продуктивності та ефективності робочих процесів у різних галузях промисловості та бізнесу.

Ключові слова: облік робочого часу, програмне забезпечення, база даних, інтерфейси, мова програмування C#, MSSQLServer, ADO.NET.

ABSTRACT

Shcherbyna O. O. Development of software for accounting of employees working hours at the enterprise.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2024.

In this qualification work, the process of designing and implementing a system for recording the working hours of employees is considered.

The existing software, which is popular on the international market and has a large number of customers in different countries, is analyzed.

Also, the first part describes what the analysis of the subject area is.

In the second section, the analysis of tools for the development of software for accounting of working hours of employees was carried out. Modern and efficient technologies and components were chosen.

Using the C# programming language and the MSSQLServer database, software has been developed that allows you to effectively track workers' time resources and automate accounting processes. The work includes analysis of subject area needs, database design, software development and testing. This system is important for enterprises because it allows to optimize the use of working time, improves management efficiency and helps in data collection and analysis. The results of the work can be used to improve the productivity and efficiency of work processes in various industries and businesses.

Keywords: time accounting, software, database, interfaces, programming language C#, MSSQLServer, ADO.NET.

ЗМІСТ

ВСТУП

В сучасних умовах розвитку ринкової економіки та глобалізації бізнесу, ефективне управління персоналом стає критичним фактором успішного функціонування підприємства. Одним із найважливіших аспектів управління персоналом є облік робочого часу співробітників, який забезпечує точне і справедливе розподілення ресурсів, підвищує продуктивність праці та сприяє дотриманню трудового законодавства. Метою даної роботи є розробка системи обліку робочого часу співробітників, яка дозволить автоматизувати процеси збору, обробки та аналізу даних про робочий час.

Актуальність розробки та впровадження системи обліку робочого часу співробітників на підприємстві обумовлена кількома ключовими факторами:

1) Підвищення ефективності управління персоналом:

– У сучасному бізнес-середовищі, де конкурентоспроможність визначається ефективністю використання ресурсів, автоматизація обліку робочого часу стає необхідністю. Впровадження такої системи дозволяє значно зменшити адміністративні витрати, оптимізувати робочі процеси та підвищити продуктивність праці.

2) Точність та прозорість даних

– Ручний облік робочого часу часто призводить до помилок і неточностей, що може викликати конфлікти між співробітниками та керівництвом. Автоматизована система забезпечує високу точність даних та прозорість процесів, що сприяє підвищенню довіри між працівниками та адміністрацією.

3) Дотримання законодавства:

– Сучасні трудові закони вимагають точного обліку робочого часу для забезпечення прав працівників. Система обліку робочого часу допомагає підприємству відповідати цим вимогам, знижуючи ризик юридичних проблем.

4) Аналіз та звітність:

– Здатність автоматично генерувати звіти і аналізувати дані дозволяє

керівництву приймати обґрунтовані рішення, що сприяє покращенню стратегічного планування і управління людськими ресурсами.

5) Інтеграція з іншими системами:

– Впровадження системи обліку робочого часу дозволяє інтегрувати її з іншими інформаційними системами підприємства, створюючи єдиний інформаційний простір. Це спрощує процеси управління та підвищує загальну ефективність підприємства.

6) Покращення планування та прогнозування:

– Зібрані дані дозволяють більш точно планувати робочі графіки, прогнозувати потреби у персоналі та розробляти ефективні стратегії управління людськими ресурсами, що сприяє зменшенню витрат і підвищенню продуктивності.

Для досягнення поставленої мети перед кваліфікаційною роботою ставляться наступні завдання:

1) Аналіз вимог користувачів (адміністраторів, керівників, співробітників) для визначення їхніх потреб та вимог щодо функціоналу системи обліку відпрацьованого часу.

2) Визначення функцій та можливостей системи на основі аналізу вимог користувачів, основних етапів робочого процесу в системі.

3) Проектування інтерфейсу:

– розробити інтерфейс системи з урахуванням зручності використання та естетичного оформлення;

– розробити макети інтерфейсу для різних типів користувачів (адміністратори, керівники, співробітники).

4) Розробка функціоналу:

– реалізувати можливість індивідуальних налаштувань для кожного користувача (мова інтерфейсу, формат відображення даних тощо);

– розробити систему керування правами доступу та рівнями доступу для забезпечення безпеки даних.

5) Тестування:

– провести тестування розробленої системи на різних етапах розробки;

– виявити та виправити помилки та недоліки під час тестування;

– впровадження та навчання персоналу.

Сучасний рівень розвитку інформаційних технологій дозволяє створювати високоефективні системи обліку робочого часу, які можуть інтегруватися з іншими системами управління підприємством, такими як ERP, HRM та інші. Існуючі на ринку рішення, такі як ADP Workforce Now, BambooHR, Kronos Workforce Dimensions та UKG Pro, надають широкий спектр функціональних можливостей для обліку та аналізу робочого часу. Проте, кожна з цих систем має свої особливості, переваги та недоліки, що обумовлює необхідність розробки власного рішення, яке буде враховувати специфіку конкретного підприємства.

Розробка системи обліку робочого часу співробітників передбачає використання сучасних технологій програмування, таких як мова програмування C#, реляційна система управління базами даних Microsoft SQL Server та платформа для доступу та управління даними ADO.NET. Ці технології забезпечують високу продуктивність, масштабованість та надійність розроблюваної системи.

Розв'язання поставленого завдання має велике значення для предметної області управління персоналом, оскільки дозволяє підвищити ефективність управління, знизити витрати, підвищити точність та прозорість даних, а також забезпечити дотримання трудового законодавства. Взаємозв'язок даної роботи з іншими дослідженнями полягає у використанні сучасних методів і підходів до автоматизації бізнес-процесів та інтеграції різних інформаційних систем для створення єдиного інформаційного простору підприємства.

Таким чином, розробка системи обліку робочого часу співробітників є актуальним і важливим завданням, що сприятиме підвищенню ефективності управління персоналом та загальної продуктивності підприємства.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з найменувань, додатків.

Обсяг роботи 98 сторінок, містить 25 рисунків та 12 таблиць.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Мета та доцільність створення програмного забезпечення для обліку робочого часу працівників

Аналіз та характеристика об'єкта проектування є ключовим етапом в розробці будь-якої системи обліку відпрацьованого часу з можливістю налаштування та персоналізації.

Системи обліку відпрацьованого часу грають важливу роль у оптимізації робочих процесів на підприємстві.

Вони дають можливість керівництву аналізувати робочі графіки співробітників та витрати ресурсів на них, що дозволяє виявляти ефективні та неефективні графіки роботи, переліки робочих годин, коли потрібні більше працівників, і навпаки. Оптимізація графіків дає можливість підприємствам знизити витрати на персонал та збільшити продуктивність.

На основі даних про відпрацьований час і вимог роботи системи обліку можуть допомогти керівництву в плануванні ресурсів і завдань, що дозволяє розподіляти завдання між працівниками ефективно та забезпечувати оптимальне використання часу кожного співробітника.

Аналіз даних обліку робочого часу дає змогу виявляти проблеми з продуктивністю та шукати шляхи їх вирішення. Це може включати ідентифікацію періодів, коли співробітники працюють неефективно, або виявлення проблем з організацією робочих процесів. Шляхи вирішення таких проблем можуть включати перерозподіл завдань, надання додаткової підтримки або навіть внесення змін у робочі процеси.

Такі системи полегшують керівництву відслідковувати прогрес проектів та завдань, враховуючи час, витрачений на їх виконання, що дозволяє вчасно виявляти можливі затримки або проблеми з виконанням завдань і приймати вчасні заходи для їх вирішення.

На основі даних обліку робочого часу можна проводити аналіз ефективності бізнес-процесів, що дає змогу виявляти та усувати проблемні аспекти в роботі підприємства, а також впроваджувати поліпшення для оптимізації продуктивності та якості роботи.

Облік відпрацьованого часу дозволяє точно розрахувати заробітну плату співробітників. Це включає оплату за відпрацьовані години, наднормові години, нічні зміни, вихідні дні та інші види роботи, які можуть мати різні тарифні ставки. Точний облік робочого часу зменшує ймовірність помилок у розрахунках, що, у свою чергу, підвищує задоволеність співробітників і знижує ризик трудових спорів.

Законодавство багатьох країн встановлює чіткі правила щодо робочого часу, наднормових годин, перерв, відпусток і відпочинку. Системи обліку робочого часу допомагають підприємствам дотримуватися цих вимог, зменшуючи ризик юридичних проблем, штрафів та інших санкцій. Вони забезпечують автоматичний контроль за дотриманням законодавчих норм, що значно полегшує роботу кадрових служб.

Системи обліку робочого часу сприяють підвищенню дисципліни серед співробітників, оскільки вони знають, що їхній робочий час відстежується. Це може зменшити кількість запізнень, передчасних відходів з роботи та інші форми неефективного використання робочого часу. Крім того, прозорість у обліку робочого часу сприяє справедливості в розрахунку заробітної плати та винагород, що підвищує мотивацію співробітників.

Прозорий і точний облік робочого часу сприяє підвищенню рівня довіри між співробітниками та керівництвом. Співробітники можуть бути впевнені, що їхня праця оцінюється справедливо, а керівництво може мати достовірні дані для прийняття управлінських рішень. Це покращує загальний моральний стан у колективі та сприяє створенню позитивної робочої атмосфери.

Зі впровадженням віддаленої роботи та гнучких графіків, підприємствам потрібно мати інструменти для ефективного управління робочим часом, незалежно від місця знаходження співробітників. Сучасні системи обліку

робочого часу надають такі можливості.

Сучасні системи обліку робочого часу дозволяють автоматизувати рутинні процеси, що знижує ймовірність помилок і підвищує загальну продуктивність підприємства. Це особливо важливо в умовах зростаючої конкуренції, де ефективність кожного співробітника має велике значення.

Мета проекту – розробка програмного забезпечення обліку відпрацьованого часу співробітників з можливістю налаштування та персоналізації для підвищення ефективності управління персоналом на підприємстві.

1.2 Огляд існуючого програмного забезпечення для автоматизації обліку відпрацьованого часу на підприємстві

Звичайно, на даний час існує багато систем, спеціально розроблених для ведення обліку часу роботи працівників на підприємстві. Приклади функціональностей та можливостей, які може надавати така система:

– дозволяє працівникам реєструвати свій час приходу та виходу з роботи, а також перерви на обід та інші відпустки чи відсутності (реєстрація часу);

– може автоматично визначати робочий час працівників на основі даних, введених ними або збережених в системі за допомогою інших методів (автоматичний облік часу);

– може створювати різні звіти та аналітику щодо відпрацьованого часу працівників, такі як зведені звіти по місяцям, тижням, дням, а також звіти про відсутності, перерви і т.д (генерація звітів і аналітики);

– може бути інтегрована з іншими системами управління персоналом, бухгалтерськими системами та іншими програмами для обміну даними і автоматизації процесів (інтеграція з іншими системами);

– забезпечує високий рівень захисту і конфіденційності даних про відпрацьований час працівників, зокрема шляхом шифрування даних та

обмеженням доступу до них за правами користувачів (захист і конфіденційність даних).

Такі інформаційні системи дозволяють ефективно вести облік часу роботи працівників на підприємстві, спрощуючи процеси обліку та забезпечуючи точність і надійність даних.

Далі наведено приклади систем, що є популярними на міжнародному ринку та мають велику кількість клієнтів у різних країнах. Вони можуть бути налаштовані для відповідності потребам різних галузей та розмірів підприємств.

ADP Workforce Now – це комплексна платформа управління персоналом, яка включає в себе рішення для обліку робочого часу. Основні функції, пов'язані з обліком робочого часу (рис. 1.1).

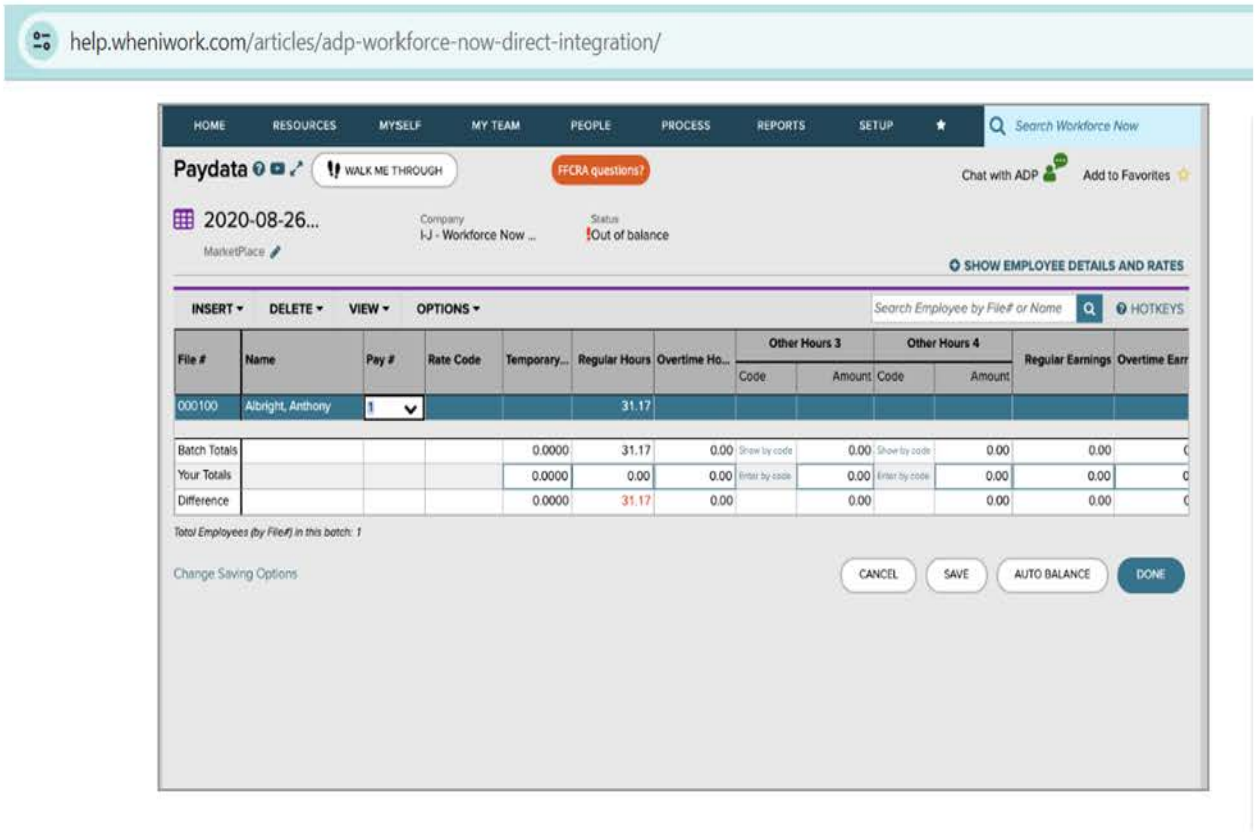


Рисунок 1.1 – Інтерфейс ADP Workforce Now

– дозволяє працівникам реєструвати свій робочий час шляхом введення

даних про початок та закінчення робочого дня, а також про перерви і відпустки.

– можна створювати гнучкі графіки робочого часу для своїх співробітників, враховуючи їхні робочі години, перерви та інші параметри;

– дозволяє відстежувати відвідуваність працівників, включаючи дати та часи їх приходу та виходу;

– дозволяє керувати відпустками та іншими видами відсутності працівників, автоматично оновлюючи їх робочий графік і розраховуючи відповідні оплати;

– надає широкий спектр звітів та аналітичних інструментів для аналізу робочого часу та ефективності персоналу;

– може бути інтегрована з іншими програмами управління персоналом, бухгалтерськими системами та іншими програмами для обміну даними та автоматизації процесів.

BambooHR – це також інтегрована система управління персоналом, яка включає в себе аналогічні функції обліку робочого часу: реєстрація та графіки робочого часу; відпустки та відсутність; звіти та аналітика; інтеграція з іншими системами (рис. 1.2).

Порівняємо BambooHR і ADP Workforce Now за декількома ключовими аспектами.

Функціонал обліку робочого часу:

– BambooHR має базовий функціонал обліку робочого часу, такий як реєстрація часу та відпусток, графіки робочого часу та деякі аналітичні звіти;

– ADP Workforce Now надає розширений функціонал, який включає у себе реєстрацію часу, графіки робочого часу, відпустки, відпустки за власний рахунок, хвороби та більш розвинені звіти та аналітику.

Інтерфейс користувача:

– інтерфейс користувача BambooHR відомий своєю простотою та інтуїтивною навігацією, що робить його зручним для користувачів будь-якого рівня;

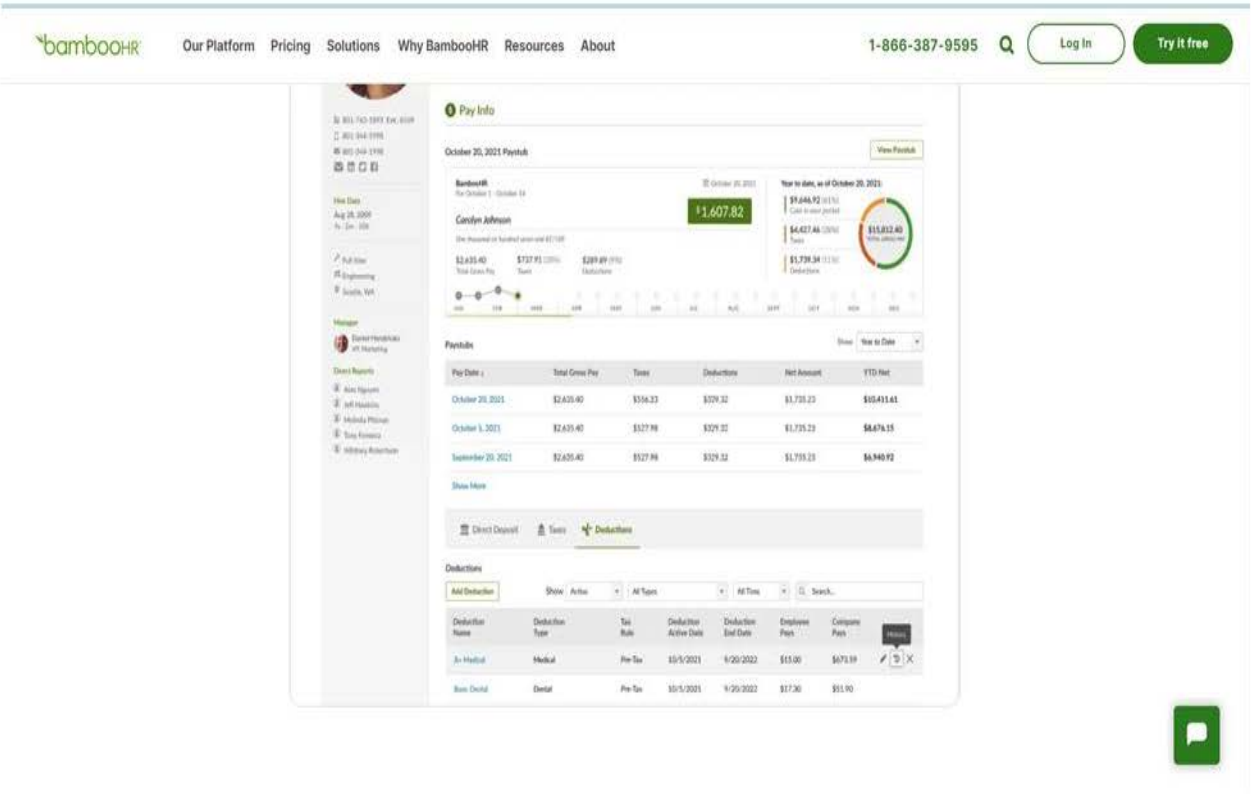


Рисунок 1.2 – Інтерфейс BambooHR

– інтерфейс користувача ADP Workforce Now може бути більш складним через розширений функціонал, але він все ж старається забезпечити зручність користування.

Ціна:

– BambooHR зазвичай має більш доступні ціни, особливо для менших підприємств;

– ADP Workforce Now може бути витратнішим, особливо для підприємств з більшим числом співробітників або потребами у складніших функціях.

Масштабованість та інтеграція:

– BambooHR часто використовується меншими та середніми підприємствами, і вона може бути менш масштабованою порівняно з ADP Workforce Now, інтеграція з іншими системами також може бути обмеженою;

– ADP Workforce Now часто використовується великими підприємствами, і вона має більше можливостей для масштабування та

інтеграції з іншими системами.

Отже, вибір між BambooHR і ADP Workforce Now буде залежати від конкретних потреб підприємства, бюджету та рівня необхідної складності.

Інтерфейс Kronos Workforce Dimensions Kronos Workforce Dimensions / UKG Dimensions (discontinued), що також є інтегрованою системою управління персоналом, наведено на рисунку 1.3. Ось деякі основні функції, пов'язані з обліком робочого часу в цій системі:

– співробітники можуть реєструвати свій робочий час через різноманітні способи, включаючи станції реєстрації часу, мобільні додатки, вебінтерфейс та інші., що може включати в себе функції розпізнавання обличчя, сканування відбитків пальців, карток доступу або PIN-кодів;

– адміністратори можуть створювати та керувати графіками робочого часу для співробітників, що включає встановлення робочих годин, перерв та інших параметрів;

– система дозволяє автоматизувати багато аспектів управління робочим часом, включаючи розподіл робочих годин, розрахунок заробітної плати, розподіл відпусток тощо;

– система надає можливість відстежувати відвідуваність працівників, включаючи дати та часи їх приходу та виходу;

– система надає широкий спектр звітів та аналітичних інструментів для аналізу робочого часу та ефективності персоналу (можна створювати звіти про відвідуваність, відпустки, робочі години та багато іншого);

– система може бути інтегрована з іншими програмами управління персоналом, бухгалтерськими системами та іншими програмами для обміну даними та автоматизації процесів.

Інтерфейс останньої у запропонованому розгляді системи, SAP SuccessFactors, наведено на рисунку 1.4. Ця система також надає рішення для реєстрації робочого часу, розрахунку зарплати, відпусток, відсутностей та інших функцій. Вона має гнучкі налаштування та може бути інтегрована з

іншими системами.

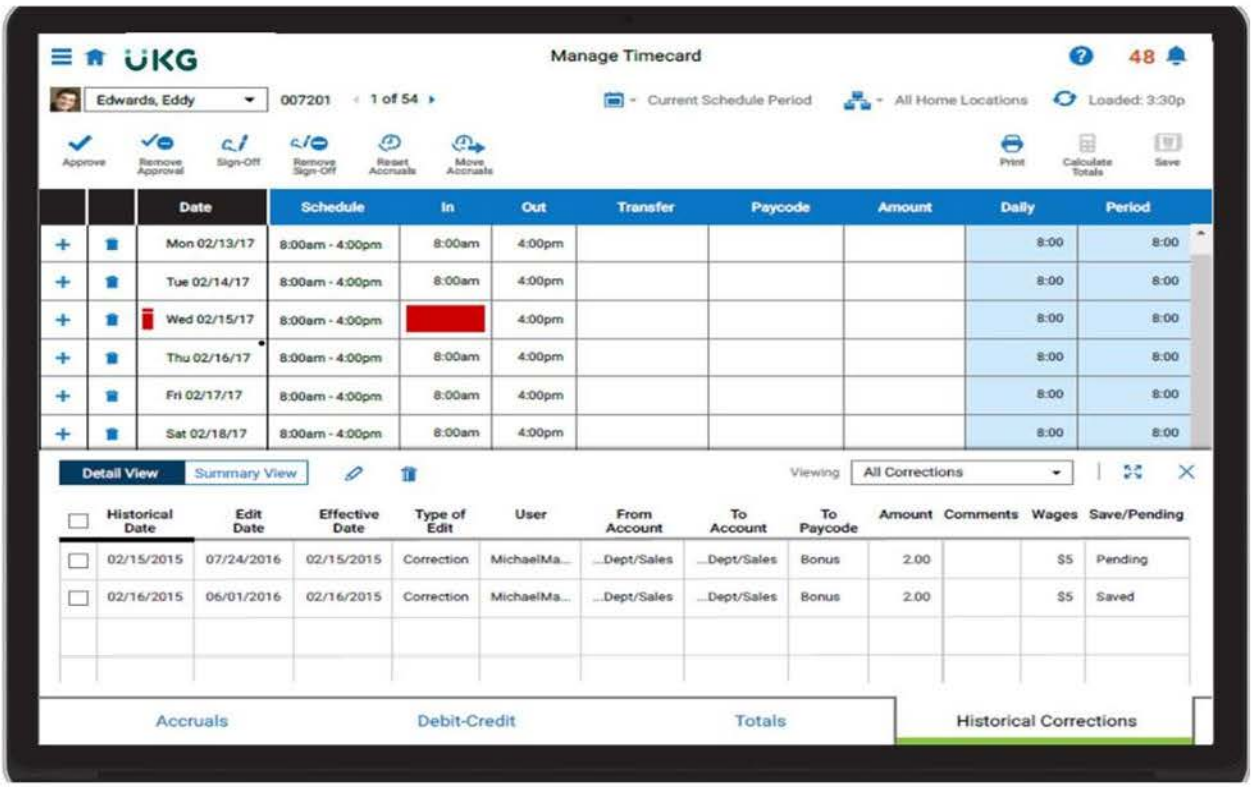
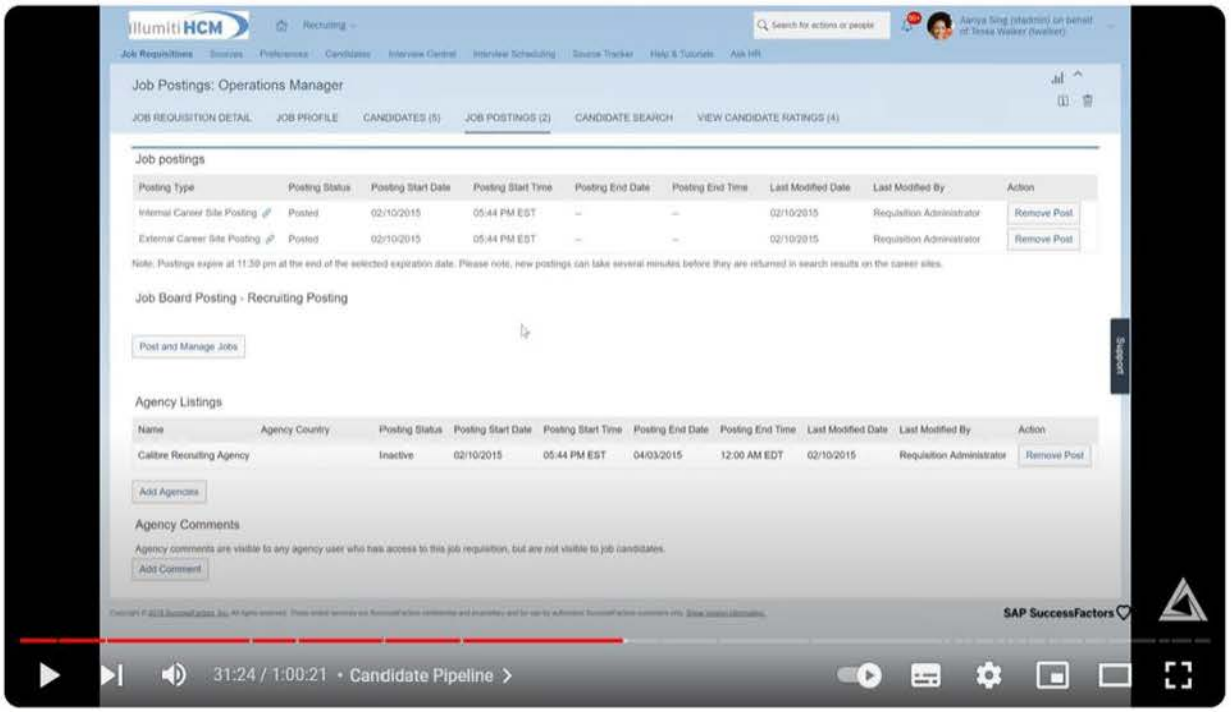


Рисунок 1.3 – Інтерфейс Kronos Workforce Dimensions Kronos Workforce Dimensions / UKG Dimensions (discontinued)



Welcome to SAP SuccessFactors Recruiting

Рисунок 1.4 – Інтерфейс SAP SuccessFactors

Якщо порівняти ці системи, то вони мають ряд відмінностей у функціоналі та підходах до управління робочим часом.

Кожна з цих систем має свою власну гнучкість налаштувань. Деякі можуть бути більш налаштовані під конкретні потреби підприємства, а інші можуть мати більш стандартизований підхід.

Усі ці системи мають свої унікальні інтерфейси користувача. Інтерфейси можуть відрізнятися за структурою, дизайном та взаємодією з користувачем.

Деякі системи, такі як Kronos Workforce Dimensions, можуть бути більш складними та масштабованими, призначеними для великих підприємств з великою кількістю співробітників, тоді як інші, наприклад BambooHR, можуть бути призначені для менших компаній або для більш специфічних потреб.

Рівень інтеграції з іншими системами може різнитися. Деякі системи можуть мати більш широкі можливості для інтеграції з іншими програмами та сервісами, що спрощує обмін даними та автоматизацію бізнес-процесів.

Вартість і рівень витрат на впровадження та підтримку цих систем можуть значно відрізнятися. Деякі системи можуть бути більш доступними для менших підприємств з обмеженим бюджетом, тоді як інші можуть бути більш витратними, але мати більше функціоналу.

Ці відмінності слід враховувати при виборі системи для управління робочим часом, оскільки вони можуть вплинути на те, яка система найкраще підходить для вашого підприємства та його

1.3 Аналіз предметної області

Аналіз предметної області включає в себе дослідження всіх аспектів, пов'язаних з функціонуванням системи, її оточенням та вимогами. Основні етапи включають:

1) визначення, що включає система і що знаходиться за її межами – визначення меж предметної області;

2) визначення функціональних та нефункціональних вимог до системи –

збір та аналіз вимог;

3) документування поточних процесів і виявлення можливостей для покращення – вивчення бізнес-процесів;

4) визначення сутностей, атрибутів і відношень – аналіз даних;

5) визначення основних функцій системи і їх взаємодії – розробка функціональної моделі;

6) визначення сутностей, атрибутів та їхніх відношень – розробка об'єктної моделі.

Функціональний аналіз предметної області – це процес вивчення та моделювання всіх функцій і процесів, що відбуваються в межах цієї області, з метою зрозуміти, як вони взаємодіють та як їх можна оптимізувати. У контексті обліку робочого часу, функціональний аналіз включає визначення основних функцій, їх підфункцій, а також взаємозв'язків між ними.

Функціональна модель предметної області за допомогою методології IDEF0 – це структурований спосіб опису і візуалізації функцій та процесів в межах певної предметної області. IDEF0 (Integration Definition for Function Modeling) використовується для моделювання та аналізу функціональних аспектів системи, що дозволяє чітко визначити, які функції виконує система, як вони взаємодіють і які ресурси для цього використовуються.

Розглянемо основні компоненти IDEF0.

Блоки (Functions) представляють функції або процеси, що виконуються в системі. Кожен блок відповідає за певну дію або набір дій.

Стрілки (Arrows):

– вхідні (Inputs): дані або матеріали, що використовуються функцією для виконання завдання;

– вихідні (Outputs): результати виконання функції;

– управління (Controls): інформація або умови, що керують виконанням функції;.

– механізми (Mechanisms): інструменти або ресурси, за допомогою яких виконується функція.

Контекстна діаграма (A-0) – найвищий рівень діаграми, яка описує основну функцію предметної області в загальному вигляді.

Декомпозиція – розбиття високорівневих функцій на більш детальні підфункції для глибшого аналізу.

Процес побудови функціональної моделі за допомогою IDEF0:

1) Визначення основної функції (A-0) – створення контекстної діаграми, що показує основну функцію системи в цілому.

2) Ідентифікація основних функцій (A0) – визначення ключових функцій, що складають основну функцію. Ці функції будуть представлені у вигляді блоків на діаграмі.

3) Декомпозиція функцій – розбиття кожної основної функції на підфункції для більш детального аналізу.

4) Визначення взаємозв'язків – визначення, які вхідні дані, вихідні дані, управління та механізми потрібні для кожної функції.

5) Створення діаграм – графічне зображення функцій і їх взаємозв'язків за допомогою блоків і стрілок.

Об’єктна модель зосереджується на структурі даних і відносинах між різними елементами предметної області. Вона використовується для моделювання об’єктів, їхніх властивостей та взаємозв’язків, що є основою для створення бази даних та інших компонентів системи.

Основні цілі:

1) Ідентифікація сутностей – модель визначає основні сутності (об’єкти) предметної області та їх атрибути.

2) Визначення відносин між сутностями – описує, як сутності взаємодіють одна з одною, включаючи асоціації, агрегації та композиції.

3) Створення логічної структури даних – модель забезпечує логічне представлення структури даних, що буде використовуватися при розробці бази даних.

4) Забезпечення узгодженості – допомагає забезпечити узгодженість даних і підтримувати цілісність інформації в системі.

Моделі предметних областей повинні відповідати ряду вимог для забезпечення їх ефективності, точності та корисності у процесі розробки інформаційних систем. Основні вимоги включають:

1) Точність і адекватність

Точність: моделі повинні точно відображати реальні бізнес-процеси, сутності та їхні відносини в предметній області. Вони мають бути засновані на достовірних і перевірених даних.

Адекватність: моделі повинні відповідати специфічним вимогам проекту і відображати сутність досліджуваної предметної області. Вони мають бути досить детальними, щоб охопити всі аспекти предметної області, але не надто складними.

2) Повнота

Повнота: моделі повинні включати всі важливі елементи та відносини, необхідні для розуміння предметної області. Жодні важливі компоненти не повинні бути упущені.

Забезпечення контексту: моделі повинні відображати контекст, в якому функціонують процеси та сутності, включаючи зовнішні взаємодії та впливи.

3) Зрозумілість і однозначність

Зрозумілість: моделі повинні бути зрозумілими для всіх зацікавлених сторін, включаючи бізнес-користувачів, розробників, аналітиків і керівників проектів. Використання стандартизованих нотацій і простих для розуміння діаграм сприяє цьому.

Однозначність: моделі повинні уникати неоднозначностей і двозначностей. Кожний елемент і відношення повинні мати чітке визначення.

4) Консистентність

Консистентність: усі частини моделі повинні бути узгодженими між собою. Не повинно бути суперечностей між різними елементами моделі або між різними моделями (наприклад, між функціональною та об’єктною моделями).

5) Модульність і масштабованість

Модульність: моделі повинні бути розділені на логічні модулі або частини, які можна аналізувати та розробляти окремо. Це полегшує розуміння та управління складними системами.

Масштабованість: моделі повинні підтримувати можливість розширення та модифікації без значних змін у їхній структурі. Це забезпечує адаптивність системи до змін у бізнес-процесах або вимогах.

6) Відтворюваність і верифікованість

Відтворюваність: моделі повинні бути достатньо детальними, щоб інші розробники могли відтворити їх на основі наявної інформації.

Верифікованість: моделі повинні підлягати перевірці та валідації. Це включає можливість тестування моделей на відповідність реальним даним і вимогам.

7) Інтегративність

Інтегративність: моделі повинні підтримувати інтеграцію з іншими системами та моделями. Це включає можливість взаємодії з існуючими системами, базами даних та іншими інформаційними ресурсами.

8) Відповідність стандартам

Відповідність стандартам: використання загальноприйнятих стандартів і нотацій (наприклад, UML, BPMN, IDEF0) забезпечує зрозумілість і сумісність моделей. Це полегшує спілкування між різними командами і зацікавленими сторонами.

1.4 Висновки до першого розділу. Постановка завдань дослідження

У першому розділі було описано мету та актуальність створення програмного забезпечення для обліку робочого часу працівників на підприємстві.

Зазначено, що підприємства різних галузей (виробничі, сервісні, IT-компанії тощо) мають потребу в ефективному обліку робочого часу співробітників.

Визначено наступні потреби підприємства:

– точний облік робочого часу для розрахунку заробітної плати;

– автоматизація обліку;

– забезпечення дотримання трудового законодавства;

– оптимізація управлінських процесів;

– інтеграція з іншими інформаційними системами (HR-системи, бухгалтерія, ERP тощо);

– зручний доступ до даних для співробітників та керівників.

Проаналізовано існуюче програмне забезпечення, що є популярним на міжнародному ринку та має велику кількість клієнтів у різних країнах.

Також у першій частині описано, що представляє собою аналіз предметної області.

Мета проекту – розробка програмного забезпечення обліку відпрацьованого часу співробітників з можливістю налаштування та персоналізації для підвищення ефективності управління персоналом на підприємстві.

Для досягнення поставленої мети перед кваліфікаційною роботою ставляться наступні завдання:

1) Аналіз вимог користувачів (адміністраторів, керівників, співробітників) для визначення їхніх потреб та вимог щодо функціоналу системи обліку відпрацьованого часу.

2) Визначення функцій та можливостей системи на основі аналізу вимог користувачів, основних етапів робочого процесу в системі.

3) Проектування інтерфейсу:

– розробити інтерфейс системи з урахуванням зручності використання та естетичного оформлення;

– розробити макети інтерфейсу для різних типів користувачів (адміністратори, керівники, співробітники).

4) Розробка функціоналу:

– реалізувати можливість індивідуальних налаштувань для кожного

користувача (мова інтерфейсу, формат відображення даних тощо);

– розробити систему керування правами доступу та рівнями доступу для забезпечення безпеки даних.

5) Тестування:

– провести тестування розробленої системи на різних етапах розробки;

– виявити та виправити помилки та недоліки під час тестування;

– впровадження та навчання персоналу.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ РОБОЧОГО ЧАСУ ПРАЦІВНИКІВ

2.1 Вибір програмних засобів для реалізації проекту

Для розробки програмного забезпечення обліку робочого часу працівників будуть використані наступні технології та компоненти:

1) мова програмування C#, що є однією з найпопулярніших мов для розробки програмного забезпечення;

2) Microsoft SQL Server – потужна реляційна система управління базами даних;

3) ActiveX Data Objects for .NET (ADO.NET) – платформа для доступу та управління даними в .NET додатках.

2.2 Мова програмування C#

C# (C-Sharp) – це сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона була представлена в 2000 році і відтоді постійно розвивається, ставши однією з найпопулярніших мов для розробки програмного забезпечення.

Мова програмування C# використовується для розробки різних типів ПЗ [1 – 5].

1) Вебдодатки: C# використовується для розробки вебдодатків за допомогою технологій ASP.NET і ASP.NET Core. Це можуть бути корпоративні вебдодатки, онлайн-магазини, соціальні мережі, блоги тощо.

2) Десктопні додатки: C# використовується для створення десктопних додатків за допомогою технологій Windows Forms, WPF (Windows Presentation Foundation) та UWP (Universal Windows Platform). Це можуть бути програми для операційних систем Windows для різних потреб, від офісних програм до ігор.

3) Мобільні додатки: C# використовується для розробки мобільних додатків для платформ iOS та Android за допомогою фреймворків Xamarin та Xamarin.Forms. Це дозволяє розробникам створювати крос-платформені додатки на C# для різних мобільних пристроїв.

4) Ігрова розробка: C# використовується для розробки ігор за допомогою фреймворків та движків, таких як Unity. Unity дозволяє створювати ігри для різних платформ, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність.

5) Системне програмування: Хоча C# зазвичай використовується на вищенаведених рівнях додатків, вона також може бути використана для системного програмування, особливо на платформі Windows, де вона може використовувати різні API та функції ОС.

6) Обробка даних та аналітика: C# може бути використаний для розробки програм для обробки даних, аналізу та візуалізації. Це можуть бути програми для роботи з базами даних, обробки великих обсягів даних, створення звітів та інші аналітичні програми.

Загалом, C# є досить універсальною мовою програмування, яка може бути використана для розробки різноманітних програмних застосунків у різних сферах індустрії.

Одною із основних характеристик C# є те, що це об’єктно-орієнтована мова програмування.

Об'єктно-орієнтоване програмування (ООП) – це парадигма програмування, яка використовує «об’єкти» для представлення даних та методів їх обробки. C# була розроблена з урахуванням принципів ООП, що робить її потужною мовою для розробки складних і масштабованих програм. Основні концепції ООП в C# включають інкапсуляцію, наслідування, поліморфізм і абстракцію.

Інкапсуляція – це механізм обмеження доступу до деяких компонентів об'єкта та захисту стану об’єкта від некоректного використання. Це досягається за допомогою модифікаторів доступу (public, private, protected,

internal).

Наслідування дозволяє створювати нові класи на основі вже існуючих, що забезпечує повторне використання коду і зручність у його модифікації. Клас, що наслідується, називається базовим (parent class, superclass), а новий клас – похідним (child class, subclass).

Поліморфізм дозволяє об'єктам різних класів оброблятись через один і той самий інтерфейс, забезпечуючи можливість виклику методів похідних класів через посилання на базовий клас. У C# поліморфізм реалізується за допомогою віртуальних методів (virtual methods), абстрактних методів (abstract methods) та інтерфейсів.

Абстракція приховує складність системи, дозволяючи зосередитись на основних функціях об'єктів, абстрагуючи їхні внутрішні реалізації. Це досягається за допомогою абстрактних класів і інтерфейсів, які визначають загальні характеристики та поведінку, не вказуючи їхню конкретну реалізацію.

Інкапсуляція захищає дані, наслідування забезпечує повторне використання коду, поліморфізм дозволяє створювати гнучкі інтерфейси, а абстракція спрощує роботу з об'єктами, приховуючи їх складність. Разом ці концепції роблять C# однією з найбільш зручних і потужних мов для розробки ПЗ.

Ще однією характеристикою мови є сильна типізація. Тобто типи всіх змінних та виразів визначаються під час компіляції і не можуть змінюватися в процесі виконання програми, що допомагає виявляти і запобігати помилкам, пов'язаним з неправильним використанням даних, ще до запуску програми.

Основним аспектом сильної типізації є статична типізація. Усі типи даних у C# повинні бути відомі на етапі компіляції. Це означає, що компілятор перевіряє типи всіх змінних і значень під час компіляції коду. Приклад статичної типізації розглянуто на рисунку 2.1.

```
// Приклад статичної типізації
int age = 25;
string name = "Alice";
bool isEmployee = true;

// age = "Alice"; // Помилка компіляції: неможливо присвоїти значення типу string змінній
```

Рисунок 2.1 – Приклад статичної типізації

У C# є два основні типи даних: типи значень (value types) і типи посилань (reference types). Типи значень зберігають свої дані безпосередньо, а типи посилань зберігають посилання на дані, які знаходяться в пам'яті (рис. 2.2).

```
// Тип значення
int value = 5;
int anotherValue = value;
value = 10;
Console.WriteLine(anotherValue); // Виведе 5, оскільки anotherValue є копією value

// Тип посилання
class Person
{
    public string Name { get; set; }
}

Person person1 = new Person { Name = "John" };
Person person2 = person1;
person1.Name = "Alice";
Console.WriteLine(person2.Name); // Виведе Alice, оскільки person2 посилається на той сами
```

Рисунок 2.2 – Приклад використання типів значень і типів посилань у C#

Сильна типізація забезпечує високий рівень надійності коду, запобігаючи багатьом поширеним помилкам, пов'язаним з неправильним використанням типів даних. Завдяки статичній типізації, перевірці типів під час компіляції, явному та неявному приведенню типів, а також підтримці

анонімних типів та виведення типів, C# надає розробникам інструменти для створення безпечних і ефективних додатків.

C# тісно інтегрований з платформою .NET, що забезпечує доступ до великої бібліотеки класів (Framework Class Library), яка використовується для розробки різних типів ПЗ. Ця інтеграція надає розробникам широкі можливості для створення ПЗ, включаючи вебдодатки, десктопні додатки, мобільні додатки, ігри, хмарні сервіси та багато іншого.

Переваги інтеграції C# з .NET:

1) Скорочення часу розробки: завдяки FCL, розробники можуть використовувати готові класи та методи, що значно скорочує час на розробку.

2) Висока продуктивність: .NET забезпечує високу продуктивність додатків завдяки оптимізованим бібліотекам та управлінню пам'яттю.

3) Крос-платформність: використовуючи .NET Core і Xamarin, можна створювати додатки, які працюють на різних операційних системах.

4) Єдність інструментарію: використання єдиної мови програмування (C#) для різних типів додатків спрощує навчання та перехід між проектами.

Ця тісна інтеграція C# з платформою .NET робить її потужним і універсальним інструментом для розробки широкого спектру ПЗ.

C# постійно оновлюється з додаванням нових функцій, які роблять розробку складних програм більш простою та ефективною.

Асинхронне програмування (async/await) дозволяє писати асинхронний код, який виконується неблокуючим чином. Ключові слова async і await дозволяють легко працювати з асинхронними операціями, такими як введення/виведення, мережеві запити та доступ до баз даних, покращуючи продуктивність і швидкодію додатків.

Лямбда-вирази відкривають можливість створювати анонімні функції, які можна передавати як параметри методам або використовувати для короткого визначення делегатів. Вони спрощують роботу з колекціями та функціональними підходами до програмування.

LINQ (Language Integrated Query) надають змогу писати запити до

колекцій об'єктів, баз даних, XML-документів та інших джерел даних прямо в коді на C#. Це забезпечує більш зрозумілий і виразний спосіб обробки даних, дозволяючи розробникам використовувати запити в стилі SQL безпосередньо в C#.

Анонімні типи дозволяють створювати об’єкти без явного визначення класу, що особливо корисно для створення тимчасових об’єктів, які використовуються для передачі даних, наприклад, у LINQ-запитах.

Pattern Matching (підтримка шаблонів) уможливлює перевіряти об’єкти на відповідність певним шаблонам і виконувати дії на основі цих шаблонів. Це спрощує роботу з умовними операторами і робить код більш читабельним та структурованим.

Tuples (кортежі) дозволяють групувати кілька значень в один об’єкт без необхідності створювати окремий клас, що зручно для повернення кількох значень з методів або для тимчасового зберігання пов'язаних даних.

Локальні функції (Local Functions) дають змогу визначати функції всередині інших функцій, що допомагає структурувати код і обмежити область видимості допоміжних функцій, які використовуються тільки в межах однієї методи.

Рекорди (Records) забезпечують зручний спосіб створення невеликих, незмінних об’єктів даних, які підтримують структурну рівність і використовують менше коду для визначення властивостей і конструкторів.

Ці сучасні конструкції роблять C# потужним інструментом для розробки складних програм, підвищуючи продуктивність розробників та спрощуючи написання чистого і підтримуваного коду.

Інтеграція з середовищем розробки Microsoft Visual Studio, яка пропонує потужні інструменти для редагування коду, налагодження, тестування та розгортання ПЗ.

Visual Studio забезпечує розширені можливості редагування коду, включаючи автозавершення, підсвічування синтаксису, рефакторинг та швидкі підказки, що допомагають писати код швидше і з меншими

помилками.

Потужні засоби налагодження дозволяють встановлювати точки зупинки, крокувати по коду, відстежувати значення змінних у реальному часі, аналізувати стек викликів та виконувати багато інших дій, необхідних для пошуку та виправлення помилок.

Visual Studio інтегрує інструменти для створення та виконання юніт-тестів, а також інших типів тестування. Це забезпечує високий рівень якості коду та стабільність ПЗ.

Інтеграція з різними платформами дозволяє легко розгортати додатки на локальних серверах, хмарних сервісах, таких як Microsoft Azure, а також на мобільних пристроях і інших середовищах.

Також є інші середовища розробки, такі як JetBrains Rider і Visual Studio Code.

Ці функції та інтеграції роблять середовища розробки потужними інструментами для створення ПЗ, значно полегшуючи життя розробникам та підвищуючи ефективність їхньої роботи.

C# продовжує розвиватися, регулярно отримуючи нові функції та вдосконалення, що відповідають сучасним вимогам до розробки ПЗ. Це робить C# привабливим вибором для розробників, які шукають потужний, гнучкий та ефективний інструмент для створення високоякісного ПЗ. Завдяки своїй універсальності та інтеграції з різноманітними середовищами розробки, C# є важливим інструментом для сучасного програмування, що сприяє продуктивності та інноваціям у сфері IT.

2.3 MSSQLServer

Microsoft SQL Server – це повнофункціональна система управління базами даних (СУБД), розроблена компанією Microsoft. Вона надає надійний, масштабований та потужний інструментарій для зберігання та обробки даних у великих корпоративних середовищах.

SQL Server базується на реляційній моделі даних, де дані зберігаються у вигляді таблиць з рядками і стовпцями. Це дозволяє ефективно організувати та керувати великими обсягами інформації –.

Реляційна структура даних – це метод організації та зберігання даних у базі даних, де дані представлені у вигляді таблиць з рядками і стовпцями. Цей підхід базується на математичній теорії реляційних відношень, яку розробив Едгар Кодд в 1970 році.

Вся інформація в базі даних організована у вигляді таблиць. Кожна таблиця має назву і складається з рядків і стовпців. Рядки таблиці представляють записи даних, а стовпці – атрибути або поля, які описують ці дані.

Ключі визначають унікальність кожного запису в таблиці. Головний ключ (Primary Key) є унікальним ідентифікатором для кожного запису і гарантує, що в таблиці немає дублікатів. Інші ключі, такі як зовнішні ключі (Foreign Keys), встановлюють зв'язки між таблицями.

Реляційна модель баз даних ґрунтується на понятті відносин між таблицями. Відносини визначаються за допомогою ключів, які встановлюють зв'язки між різними таблицями. Це дозволяє створювати складні структури даних зі зв'язками між різними сутностями.

Нормалізація – це процес розбиття таблиць на менші і більш узагальнені частини для зменшення дублікації даних і забезпечення цілісності даних. Це допомагає покращити ефективність запитів і зменшити ризик виникнення аномалій даних.

Для взаємодії з реляційною базою даних використовується мова запитів SQL (Structured Query Language). SQL надає стандартизований набір команд для вибірки, оновлення, вставки та видалення даних з таблиць.

Реляційна база даних підтримує концепцію транзакцій, які гарантують атомарність, консистентність, ізольованість та стійкість (ACID) операцій з даними. Це дозволяє забезпечити надійність та цілісність даних в системі.

Реляційна структура даних є одним з найпоширеніших та ефективних

підходів до організації та управління даними в базах даних. Вона забезпечує гнучкість, надійність та ефективність при роботі з великими обсягами інформації.

Взаємодія з базою даних SQL Server відбувається за допомогою мови запитів SQL (Structured Query Language). SQL дозволяє виконувати різноманітні операції, такі як додавання, вилучення, оновлення та вибірка даних з бази даних [6 – 10].

SQL (Structured Query Language) – це стандартизована мова програмування, яка використовується для взаємодії з реляційними базами даних. Вона дозволяє користувачам виконувати різноманітні операції з даними, такі як вибірка, вставка, оновлення та видалення, а також керувати структурою бази даних і здійснювати адміністративні операції.

SQL є стандартизованою мовою, існують різні версії стандарту SQL, такі як SQL-86, SQL-92, SQL:1999, SQL:2003 та інші. Кожна версія стандарту включає нові функції та покращення, але більшість СУБД дотримуються загальних принципів стандарту SQL.

Мова програмування надає ряд команд для взаємодії з даними в базі даних. Найбільш базові операції включають SELECT (вибірка), INSERT (вставка), UPDATE (оновлення) та DELETE (видалення).

SQL дозволяє визначати та змінювати структуру бази даних за допомогою команд CREATE, ALTER та DROP. За допомогою цих команд можна створювати нові таблиці, індекси, відносини, змінювати структуру існуючих об’єктів і видаляти їх.

Умовні оператори в SQL включають WHERE, HAVING, CASE, що дозволяє виконувати операції з даними на основі певних умов.

За допомогою різних типів з’єднань, таких як INNER JOIN, LEFT JOIN, RIGHT JOIN та FULL JOIN SQL дозволяє об’єднувати дані з різних таблиць.

SQL дозволяє групувати дані за певними характеристиками та обчислювати агрегатні функції, такі як COUNT, SUM, AVG, MIN, MAX для аналізу даних в групах.

SQL підтримує транзакції, що дозволяє виконувати групу операцій як єдину атомарну операцію. Це забезпечує цілісність даних та відновлення системи в разі виникнення помилок.

Тобто мова програмування SQL є потужним інструментом для роботи з базами даних, що використовується в різних галузях для зберігання, обробки та аналізу великих обсягів даних.

Можливості управління даними в Microsoft SQL Server включають широкий спектр інструментів і функцій, які дозволяють адміністраторам та розробникам ефективно керувати базами даних, забезпечуючи їх надійність, доступність та безпеку.

SQL Server надає можливість створювати нові бази даних з різними параметрами, такими як розмір файлів даних та журналів, кодування та настройки.

Приклад підключення до бази даних і виконання запиту наведено на рисунку 2.3.

```
using System;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString = "Server=your_server;Database=your_database;User Id=your_

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();
            Console.WriteLine("Connection Opened");

            string query = "SELECT COUNT(*) FROM your_table";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                int count = (int)command.ExecuteScalar();
                Console.WriteLine("Number of rows in your_table: " + count);
            }
        }
```

Рисунок 2.3 – Приклад використання типів значень і типів посилань у C#

Адміністратори можуть визначати та змінювати структуру баз даних, включаючи створення, модифікацію, видалення таблиць, індексів, виділення та інших об'єктів бази даних.

SQL Server забезпечує гнучкі можливості керування доступом до даних, включаючи призначення ролей, дозволів на рівні таблиць та об’єктів, а також обмеження доступу за допомогою ряду механізмів аутентифікації та авторизації.

Також MSSQLServer дозволяє створювати регулярні резервні копії баз даних та журналів транзакцій для забезпечення їх безпеки та відновлення в разі втрати даних або виникнення неполадок; надає різноманітні інструменти для моніторингу та аналізу працездатності та продуктивності баз даних. Ці інструменти дозволяють виявляти проблеми та оптимізувати використання ресурсів.

Включає різні механізми оптимізації запитів, такі як генерація планів виконання, індексація та кешування, що дозволяють підвищити продуктивність виконання запитів та зменшити навантаження на сервер баз даних.

Microsoft SQL Server веде журнали транзакцій, які дозволяють відстежувати зміни даних та відновлювати базу даних до певних моментів в минулому в разі потреби.

Також SQL Server надає різні механізми шифрування даних, такі як Transparent Data Encryption (TDE) та Always Encrypted, що дозволяють захистити дані від несанкціонованого доступу та забезпечити їх конфіденційність. На рисунку 2.4 наведено код, що увімкне TDE для бази даних з назвою «EncryptionSample». Дані в цій базі даних будуть автоматично шифруватися на рівні файлів.

Всі ці можливості дозволяють адміністраторам та розробникам ефективно керувати базами даних в Microsoft SQL Server, забезпечуючи їх надійність, доступність та безпеку. Вона відома своєю надійністю, безпекою та продуктивністю, що робить її однією з переважних платформ для розробки

додатків та управління даними.

```
-- Увімкнення Transparent Data Encryption (TDE) для бази даних
USE master;
GO
CREATE DATABASE EncryptionSample;
GO
USE EncryptionSample;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
GO
ALTER DATABASE EncryptionSample SET ENCRYPTION ON;
GO
```

Рисунок 2.4 – Увімкнення Transparent Data Encryption (TDE) для бази даних

2.4 ActiveX Data Objects for .NET

ADO.NET (ActiveX Data Objects for .NET) – це платформа, яка надає доступ до даних з різних джерел для програм, написаних на мовах .NET, таких як C#, VB.NET, інших. ADO.NET дозволяє взаємодіяти з реляційними та нереляційними даними, такими як бази даних SQL Server, Oracle, MySQL, XML-документи та інші джерела даних.

ADO.NET включає в себе різні компоненти, які дозволяють програмам взаємодіяти з даними [11 – 15].

Компонент Connection (З'єднання) встановлює з'єднання з базою даних чи іншим джерелом даних за допомогою різних параметрів підключення, таких як рядок підключення, ім'я сервера, ім'я бази даних тощо. У прикладі параметри підключення (такі як адреса сервера, назва бази даних, ім'я користувача та пароль) визначаються в конфігураційному файлі app.config або web.config, а програма отримує їх із цього файлу за допомогою ConfigurationManager (рис. 2.5).

```
using System;
using System.Data.SqlClient;
using System.Configuration;
class Program
{
    static void Main()
    {
        string connectionString = ConfigurationManager.ConnectionStrings["MyConnectionStri
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            try
            {
                connection.Open();
                Console.WriteLine("Підключення успішно встановлено до бази даних.");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка підключення: " + ex.Message);
            }
        }
    }
}
```

Рисунок 2.5 – Приклад використання параметрів підключення з конфігураційного файлу

Connection дозволяє відкривати та закривати з'єднання з джерелом даних, а також виконувати різноманітні дії, пов'язані з управлінням життєвим циклом з'єднання; забезпечує обробку винятків та помилок, пов'язаних з встановленням з'єднання, що дозволяє програмі відповідним чином реагувати на можливі проблеми з підключенням.

Загалом, компонент «Connection» є ключовим елементом в роботі з базами даних та іншими джерелами даних у середовищі ADO.NET, оскільки він забезпечує основний зв'язок між програмою та джерелом даних.

Компонент Command в ADO.NET відповідає за виконання SQL-запитів або збережених процедур до джерела даних. Цей компонент є важливим елементом для взаємодії з базою даних, оскільки він дозволяє здійснювати всі основні операції з даними, такі як вставка, оновлення, видалення та вибірка

даних.

Для створення об’єкта Command використовується клас SqlCommand, OleDbCommand, OdbcCommand, або OracleCommand, в залежності від типу бази даних. Приклад використання Command наведено на рисунку 2.6.

```
using System;
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString = "Data Source=myServerAddress;Initial Catalog=myDataBase;
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            SqlCommand command = new SqlCommand("SELECT COUNT(*) FROM MyTable", connection
            try
            {
                connection.Open();
                int count = (int)command.ExecuteScalar();
                Console.WriteLine("Кількість рядків у таблиці MyTable: " + count);
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка виконання запиту: " + ex.Message);
            }
        }
    }
```

Рисунок 2.6 – Приклад використання Command

У цьому прикладі:

1) створюється об’єкт SqlCommand з SQL-запитом SELECT * FROM MyTable;

2) відкривається з’єднання з базою даних;

3) виконується запит за допомогою ExecuteReader(), і результати читаються за допомогою об’єкта SqlDataReader;

4) якщо виникає помилка, виводиться відповідне повідомлення.

Компонент Command є невід'ємною частиною ADO.NET і забезпечує гнучкість та потужність для виконання різних операцій з даними в базі даних.

Компонент DataReader забезпечує читання потоку даних з результатів запиту без необхідності зберігання всіх даних в пам'яті.

Компонент DataAdapter використовується для заповнення DataSet (кешований набір даних) з джерела даних та оновлення даних з DataSet назад до джерела даних.

Компонент DataSet може зберігати дані з різних джерел та їхні відносини.

ADO.NET дозволяє розробникам створювати потужні та ефективні програми для роботи з різними джерелами даних у середовищі .NET.

2.5 Висновки до другого розділу

У другому розділі був проведений аналіз інструментів для розроблення програмного забезпечення обліку робочого часу працівників. Були обрані сучасні та ефективні технології та компоненти.

Використання мови програмування C# забезпечує надійну основу для створення додатків завдяки її популярності та підтримці широкого спектра інструментів.

Потужна реляційна система управління базами даних Microsoft SQL Server надає можливості для зберігання та ефективного управління даними.

Платформа ADO.NET забезпечує зручний і продуктивний доступ до даних, інтегруючись із середовищем .NET, що дозволяє розробникам легко взаємодіяти з різними джерелами даних.

Усі ці програмні засоби забезпечують надійну та ефективну основу для розробки системи обліку робочого часу, яка відповідатиме вимогам сучасного бізнесу.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОБЛІКУ ВІДПРАЦЬОВАНОГО ЧАСУ СПІВРОБІТНИКІВ

3.1 Функціональна модель предметної області обліку робочого часу працівників на підприємстві

Контекстна діаграма функціональної моделі – це візуальне представлення взаємодії між системою та її зовнішнім середовищем. Вона показує, як система взаємодіє з іншими суб'єктами, окремими модулями або зовнішніми системами. На контекстній діаграмі зображується сама система (центральний об'єкт) та зовнішні суб'єкти (зв'язки), з якими вона взаємодіє. Це допомагає зрозуміти, як система вписується у своє оточення і які обмінні процеси відбуваються між нею та іншими частинами системного середовища.

У контекстній діаграмі вхід – табель відпрацьованого часу; вихід – звіт про відпрацьований час. Завершений вигляд контекстної діаграми показано на рисунку 3.1.

Рисунок 3.1 – Контекстна діаграма

Кожна задача розбивається на складові елементи наступного рівня, і цей процес триває до тих пір, поки не буде сформована структура, яка відповідає на питання, поставлені перед створенням функціональної моделі предметної області. Кожна окрема функція моделюється окремим блоком. Кожен вищий

блок детально описується дочірньою діаграмою на більш низькому рівні. Діаграма декомпозиції першого рівня показана на рис. 3.2. У випадку, коли дії в предметній області є складними та включають низку операцій, рекомендується створювати діаграми декомпозиції другого рівня.

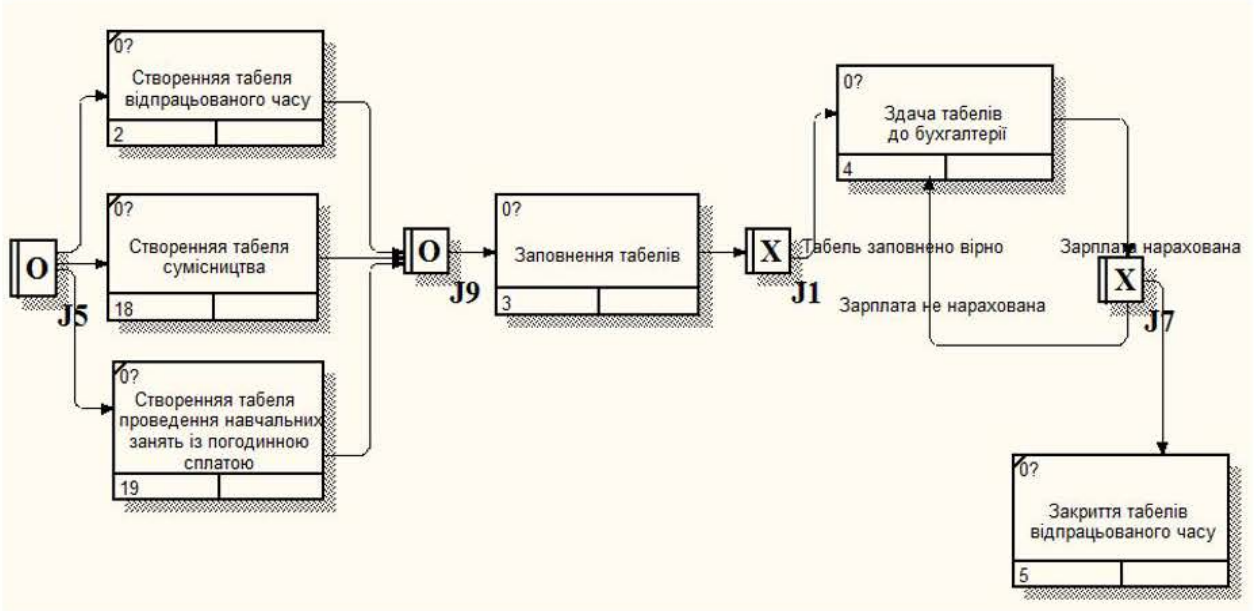


Рисунок 3.2 – Діаграма декомпозиції першого рівня

Для функціонального блоку 3 та функціонального блоку 4 необхідно створити діаграми декомпозиції другого рівня (рис. 3.3 та рис. 3.4 відповідно) для опису складових частин цих дій.

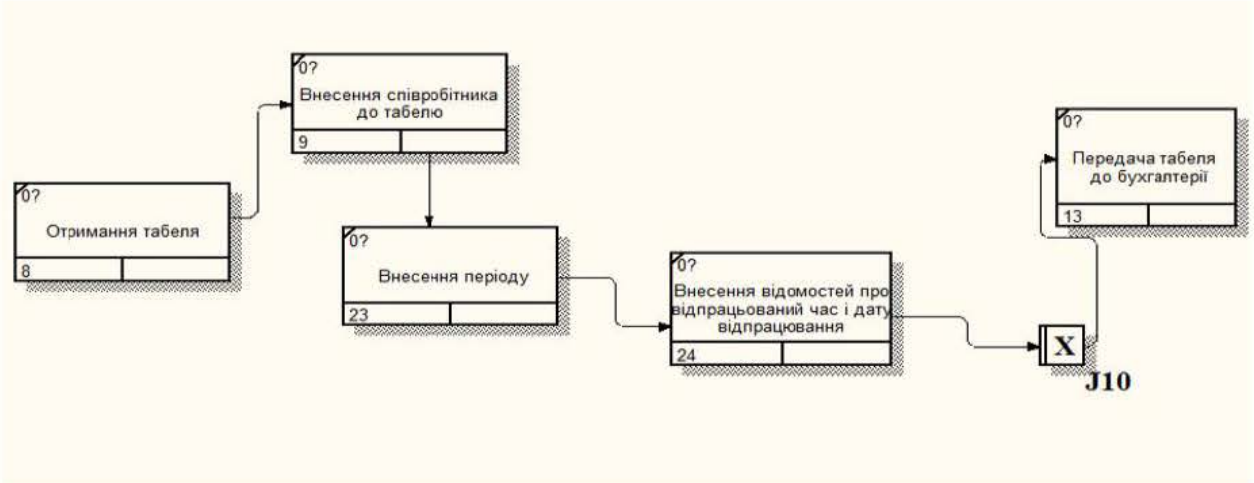


Рисунок 3.3 – Діаграма декомпозиції 2-го рівня 3-го функціонального блоку

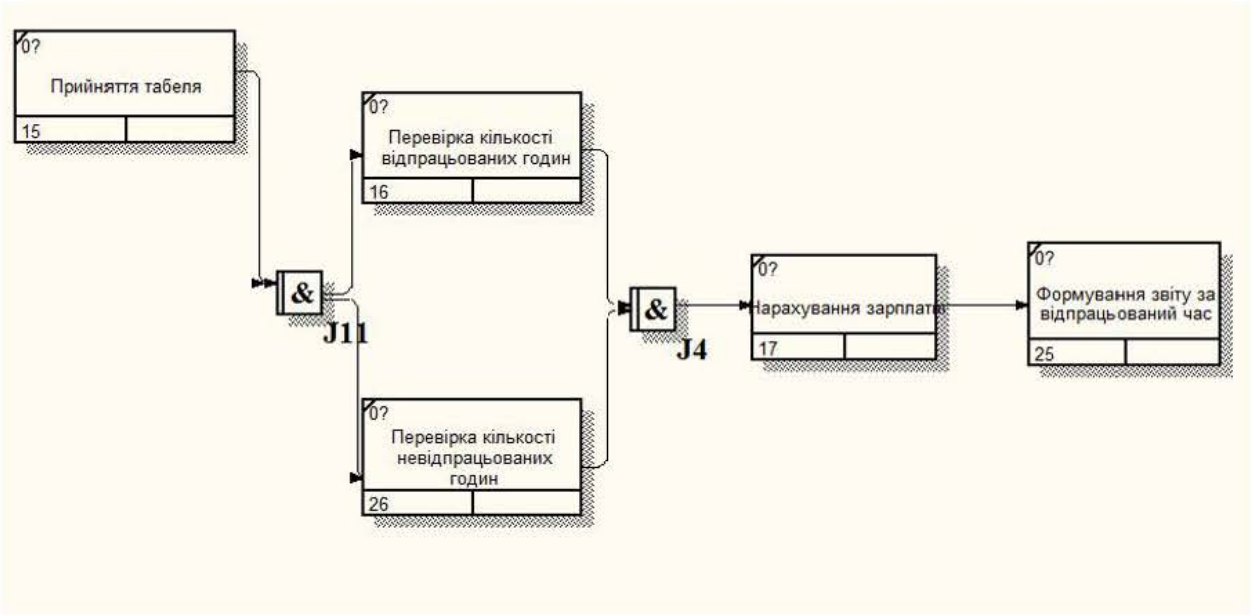


Рисунок 3.4 – Діаграма декомпозиції 2-го рівня 4-го функціонального блоку

3.2 Об’єктна модель предметної області

За допомогою об’єктно-орієнтованого методу аналізу предметної області було виявлено статичні та динамічні об’єкти, їх атрибути, взаємозв'язки і структурні особливості. Для відображення об’єктної моделі предметної області була обрана діаграма сутність-зв’язок. З урахуванням вимог розробки необхідно описати специфікацію об’єктів (таблиця 3.1) предметної області, специфікацію атрибутів сутностей (таблиця 3.2) та специфікацію зв’язків між об’єктами (таблиця 3.3).

Статичними об’єктами предметної області є співробітники, посади, відділи, позначки.

Таблиця 3.1

Специфікація сутностей предметної області обліку робочого часу працівників на підприємстві

№	Назва сутності	Описання сутності
1	Співробітник	Працівник на підприємстві
2	Посада	Статична сутність, яка описує посади, які

		обіймають співробітники
3	Відділ	Статична сутність, що описує частину установи чи підприємства
4	Позначки	Статична сутність, яка описує якісь знаки, якими відмічається присутність Співробітника в робочий час на робочому місці
5	Табель відпрацьованого часу	Документ предметної області, в якому фіксуються час, який був відпрацьований Співробітником і за допомогою якого Співробітнику нараховують заробітну платню

Таблиця 3.2

Специфікація атрибутів сутностей

№	Назва сутності	Назва атрибуту	Опис атрибуту
1	Табель відпрацьованого часу	Номер	Чисельний тип даних. Задає порядковий номер документу.
		Дата	Тип даних – дата. Задає календарну дату створення документу.
		Звітний період (з)	Символьний тип даних. Задає звітний період (з)
		Звітний період (по)	Символьний тип даних. Задає звітний період (по)
2	Співробітники у табелі	Табель відпрацьованого часу	Посилання на об’єкт «Табель відпрацьованого часу»
		Співробітник	Посилання на об’єкт «Співробітник»
3	Відпрацьований час	День місяця	Тип даних – дата. Задає день місяця
		Кількість годин	Чисельний тип даних. Задає кількість годин
		Співробітники у табелі	Посилання на об’єкт «Співробітники у табелі»
		Позначка	Посилання на об’єкт «Позначка»
4	Позначка	Назва позначки	Символьний тип даних. Задає назву позначки

5	Посада	Назва посади	Символьний тип даних. Задає назву посади
6	Відділ	Назва відділу	Символьний тип даних. Задає назву відділу
7	Співробітник	ПІБ співробтіника	Символьний тип даних. Задає ПІБ співробітника
		Табельний номер	Чисельний тип даних. Задає табельний номер співробітника
		Посада	Посилання на об’єкт «Посада»
		Відділ	Посилання на об’єкт «Відділ»

Таблиця 3.3

Зв’язки між сутностями

Назва зв’язку	Назва 1 сутності	Назва 2 сутності	Множинність зв’язку
Містить	Табель відпрацьованого часу	Співробітник	1:N
Містить	Співробітник	Посада	1:1
Містить	Співробітник	Відділ	1:1
Містить	Співробітник	Позначка	1:1

Діаграма сутність-зв'язок використовує такі графічні елементи:

1) прямокутник представляє сутність предметної області і містить назву сутності (іменник);

2) овал представляє атрибут сутності і містить назву атрибута сутності (іменник);

3) ромб відображає зв'язок між сутностями. У ромбі зазначається або назва зв'язку між сутностями (змістове дієслово), або порядковий номер.

На рисунку 3.5 наведено загальний вигляд діаграми сутність-зв'язок для розглядуваної предметної області.

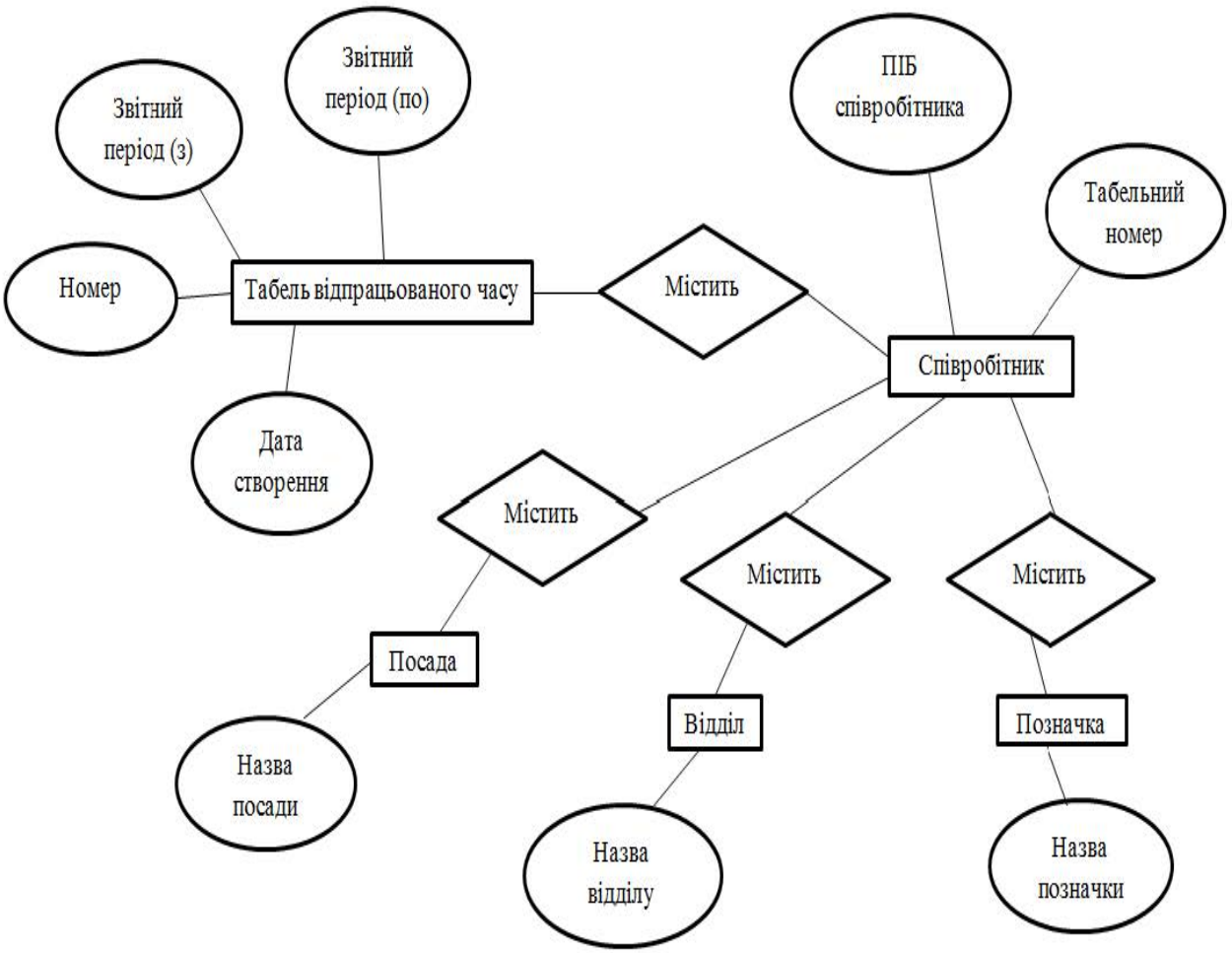


Рисунок 3.5 – Діаграма «сутність-зв’язок» предметної області

3.3 Структура меню

Система складається з нормативно-довідкової, вхідної та вихідної інформації. Для визначення компонентів системи слід провести аналіз завдання, відповідаючи на такі запитання:

1) Яку інформацію повинна надавати система для вирішення конкретної задачі у визначеній предметній області?

2) Які параметри введення використовуються для запитів до бази даних?

3) З яких документів предметної області можна отримати необхідну інформацію?

4) Яку інформацію предметної області можна вважати умовно незмінною, що можна зберегти в нормативно-довідковій інформації для

забезпечення швидкості та достовірності введення даних?

Результати динамічних запитів до бази даних складають вихідну інформацію системи. Вона включає в себе завдання, що передбачаються для створення системи. Вихідна інформація поділяється на такі складові:

– вхідні параметри запиту до БД – умови, що обмежують пошук і обсяг вибірки даних;

– результати вибірки даних із бази даних (з таблиць документів БД) ;

– автоматично розраховані результати.

Визначивши склад вихідної інформації потрібно вибрати з документообігу предметної області мінімальний набір документів, що зберігають у собі максимальну кількість даних, придатних для рішення поставленої задачі. Вхідна інформація системи – це основне джерело даних в БД, зазвичай паперові документи предметної області перетворюють до електронних документі та відносять до пункті меню АІС «Вхідна інформація» і збережені у вигляді таблиць БД. Умовно-постійні атрибути документів вхідної інформації складають основу для розробки нормативно-довідникової інформації системи. Нормативно-довідникова інформація слугує для збереження умовно незмінних даних сутностей предметної області (ПрО). Мета використання НДІ:

– скорочення часу для введення даних оператором системи;

– скорочення помилок при введенні даних;

– забезпечення цілісності даних БД.

При розробці структури довідників приймаються до уваги умови:

– одна класифікаційна група повинна мати не більш, ніж 100 рядків;

– структура довідника повинна відповідати структурі об’єкта ПрО;

– один довідник – це опис одного об’єкта предметної області.

Система обліку відпрацьованого часу повинна містити нормативно-довідникову інформацію (рис. 3.6):

1) позначки;

2) посада;

3) відділ;

4) співробітник.

До складу вхідної інформації системи необхідно віднести наступний документ предметної області – табель відпрацьованого часу.

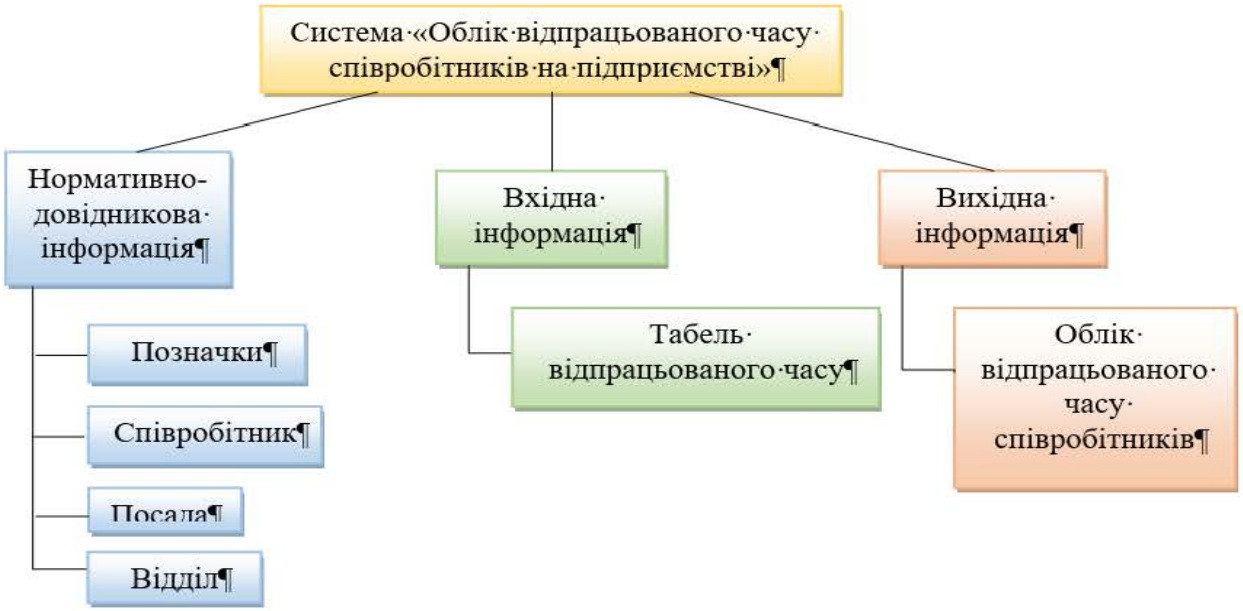


Рисунок 3.6 – Структура меню

3.4 Розробка бази даних

Розробка бази даних включає створення логічної та фізичної моделей. Логічна структура бази даних є узагальненим, табличним описом сутностей предметної області, що розробляється з урахуванням нормалізації даних та незалежно від конкретної системи управління базами даних. Логічну структуру можна представити кількома способами, найбільш інформативним з яких є повна атрибутивна модель БД. Фізична модель БД – це схема БД, розроблена в межах обраної СУБД. Оскільки вихідна інформація системи є результатом вибірок даних із БД, що генерується в залежності від запитів користувача програмного забезпечення, то в БД вихідна інформація не зберігається.

Побудова повної атрибутивної моделі БД включає в себе відображення

таблиць бази даних у вигляді таблиці, що містить 3 рядки, в яких описуються атрибути сутності:

– природна назва атрибутів сутності;

– синтетична назва атрибутів сутності;

– тип даних та розмір атрибута сутності.

В таблиці виділяються первинні та зовнішні ключі, таблиця отримує природну назву та в дужках записується назва таблиці в БД.

Нормативно-довідникова інформація АІС

Довідник «Позначки»

Таблиця 3.4

Логічна структура таблиці БД «Позначки» (Mark)

Код позначки	Назва позначки
ID_Mark	Mark_Name
I2	C10

Довідник «Посада»

Таблиця 3.5

Логічна структура таблиці бази даних «Посада» (Posada)

Код посади	Назва посади
ID_Posada	Mark_Name
I4	C50

Довідник «Відділ»

Таблиця 3.6

Логічна структура таблиці бази даних «Відділ» (Viddil)

Код відділу	Назва відділу
ID_Viddil	Viddil_Name
I4	C50

Довідник «Співробітник»

Таблиця 3.7

Логічна структура таблиці бази даних «Одиниці вимірювання» (Spivrobitnuk)

Код одиниці вимірювання	Код відділу	Код посади	Одиниця вимірювання	Табельний номер
ID_Spivrobitnuk	ID_Viddil	ID_Posada	Spivrobitnuk_Name	Tabel_Number
I4	I4	I4	C50	I4

Вхідна інформація

Таблиця 3.8

Логічна структура таблиці бази даних «Табель відпрацьованого часу» (Tabel_vc)

Код табеля	Дата створенн	Номер	Звітній період (з)	Звітній період (по)
ID_Tabel_vc	Date_create	Number	Zvit_period_z	Zvit_period_po
I8	D8	I8	D8	D8

Таблиця 3.9

Логічна структура таблиці бази даних «Співробітник у табелі» (SpivrobitnukTable)

Код співробітника у табелі	Код табеля	Код співробітника
ID_SpivrobitnukTable	ID_Tabel_vc	ID_Spivrobitnuk
I8	I8	I8

Таблиця 3.10

Логічна структура таблиці бази даних «Відпрацьований час» (VidpracTime)

Код відпрацьованого часу	Код співробітника у табелі	Код позначки	День місяця	Кількість годин
ID_ VidpracTime	ID_SpivrobitnukTable	ID_Mark	Den_mis	Kilk_god
I8	I8	I2	D8	I2

У фізичній моделі описується вся інформація про конкретні фізичні об'єкти – таблиці, колонки, індекси та збережені процедури (рис. 3.7).

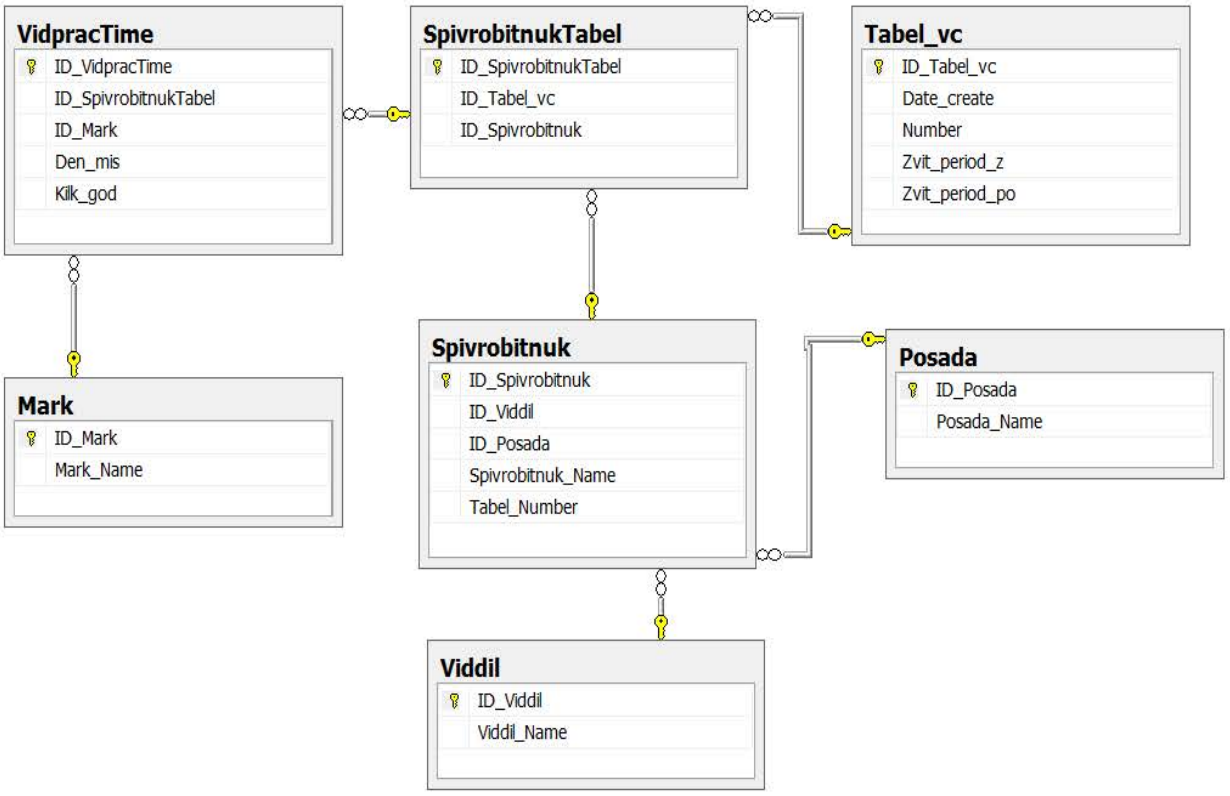


Рисунок 3.7 – Фізична модель

3.5 SQL-запити вихідної інформації

Для забезпечення можливості виконання динамічних запитів користувачів системи можна використовувати механізм запитів до представлень бази даних. Представлення є тимчасовими наборами даних, які існують тільки під час роботи з ними. Важливо розуміти, що представлення не є копією даних, а лише посиланням на реальні записи. Тобто при зміні даних у представленні змінюються також дані у фізичній таблиці, і навпаки. Представлення дозволяють повертати набори даних, які задовольняють запитам конкретних користувачів або груп. Після створення представлення до нього можна звертатися точно так само, як і до звичайної таблиці.

Представлення можуть будуватися на базі однієї або декількох таблиць, або навіть на основі інших представлень.

1) Представлення «Облік відпрацьованого часу».

SQL-запит для формування вибірки даних на основі представлення «Облік відпрацьованого часу»

```
SELECT ID_Tabel_vc, Date_create, Number, Zvit_period_z, Zvit_period_po

FROM Tabel_vc
WHERE (CONVERT(SMALLDATETIME, CONVERT(VARCHAR(10), Date_create, 101), 101) >= @Param1) AND (CONVERT(SMALLDATETIME, CONVERT(VARCHAR(10), Date_create, 101), 101) <= @Param2)
```

2) Представлення «Облік відпрацьованого часу».

SQL-запит для формування вибірки даних на основі представлення «Облік відпрацьованого часу»

```
SELECT SpivrobitnukTabel.ID_SpivrobitnukTabel,
SpivrobitnukTabel.ID_Tabel_vc, Tabel_vc.Number,
SpivrobitnukTabel.ID_Spivrobitnuk, Spivrobitnuk.Spivrobitnuk_Name
FROM SpivrobitnukTabel INNER JOIN
Tabel_vc ON SpivrobitnukTabel.ID_Tabel_vc = Tabel_vc.ID_Tabel_vc
INNER JOIN Spivrobitnuk ON SpivrobitnukTabel.ID_Spivrobitnuk = Spivrobitnuk.ID_Spivrobitnuk
WHERE (Spivrobitnuk.Spivrobitnuk_Name LIKE @Param1)
ORDER BY SpivrobitnukTabel.ID_SpivrobitnukTabel DESC
```

3) Представлення «Облік відпрацьованого часу».

SQL-запит для формування вибірки даних на основі представлення «Облік відпрацьованого часу»

```
SELECT VidpracTime.ID_VidpracTime, VidpracTime.ID_SpivrobitnukTabel,
Spivrobitnuk.Spivrobitnuk_Name, VidpracTime.ID_Mark, Mark.Mark_Name,
VidpracTime.Den_mis, VidpracTime.Kilk_god
FROM VidpracTime INNER JOIN
```

```
Mark ON VidpracTime.ID_Mark = Mark.ID_Mark INNER JOIN
SpivrobitnukTabel ON VidpracTime.ID_SpivrobitnukTabel =
SpivrobitnukTabel.ID_SpivrobitnukTabel INNER JOIN
Spivrobitnuk ON SpivrobitnukTabel.ID_Spivrobitnuk =
Spivrobitnuk.ID_Spivrobitnuk
WHERE (CONVERT(SMALLDATETIME, CONVERT(VARCHAR(10),
VidpracTime.Den_mis, 101), 101) >= @Param1) AND
(CONVERT(SMALLDATETIME, CONVERT(VARCHAR(10),
VidpracTime.Den_mis, 101), 101) <= @Param2)
ORDER BY VidpracTime.ID_VidpracTime DESC
```

3.6 Інтерфейс

Інтерфейс користувача (UI – user interface) – це сукупність інструментів, що дозволяють користувачеві взаємодіяти з програмою, включаючи всі елементи та компоненти програмного забезпечення, які впливають на спілкування користувача з ним.

У роботі використано розподіл прав доступу до даних на рівні додатку. В рамках предметної області існують 2 типи користувачів із різним рівнем доступу до даних.

Адміністратор (А) – це користувач з найвищим рівнем доступу, якому надається можливість додавати, видаляти, редагувати та здійснювати пошук даних у базі даних в межах системи.

Користувач (К) – особа з середнім рівнем доступу, якій надається можливість лише здійснювати пошук даних у базі даних в межах системи.

Операції з даними, доступні на кожній екранній формі, вказано в таблиці 3.11.

Таблиця 3.11

Визначення прав доступу до даних по видах користувачів

Програмна назва сторінки	Опис функціонального призначення сторінки	Доступ для користувачів	
		А	К
Form10.cs	авторизація	+	+
Form1.cs	Перегляд довідників	+	-
	Перегляд вхідної інформації	+	-
	Перегляд вихідної інформації	+	+
Form2.cs	Перегляд	+	-
	Додавання	+	-
	Редагування	+	-
	Пошук	+	-
Form3.cs	Перегляд	+	-
	Додавання	+	-
	Редагування	+	-
	Пошук	+	-
Form4.cs	Перегляд	+	-
	Додавання	+	-
	Редагування	+	-
	Пошук	+	-
Form5.cs	Перегляд	+	-
	Додавання	+	-
	Редагування	+	-
	Пошук	+	-
Form6.cs	Перегляд	+	-
	Редагування	+	-

	Видалення	+	-
	Пошук	+	-
Form7.cs	Додавання	+	-
Form8.cs	Пошук	+	+
	Перегляд	+	+
Form9.cs	Пошук	+	+
	Перегляд	+	+

3.7 Екранні форми

Екранні форми призначені для спрощення введення даних користувачем у базу даних. Крім того, вони можуть містити текстові пояснення, які допомагають зрозуміти, як правильно ввести дані в кожне поле, а також надати приклади заповнення цих полів. Важливо, щоб такі форми були легко доступними для перегляду й зрозумілими, оскільки це має велике значення для їхньої ефективної роботи. Наведена структура інтерфейсної частини вхідної інформації системи приведена в таблиці 3.12.

Таблиця 3.12

Структура інтерфейсної частини вхідної інформації системи

Екранна форма	Опис
Form1.cs	Екранна форма авторизації користувача
Form1.cs	Головна форма АІС
Form2.cs	Екранна форма довідника «Позначки»
Form3.cs	Екранна форма довідника «Посада»
Form4.cs	Екранна форма довідника «Відділ»
Form5.cs	Екранна форма довідника «Співробітник»
Form6.cs	Екранна форма перегляду документу «Табель відпрацьованого часу»

Form7.cs	Екранна форма додавання документу «Табель відпрацьованого часу»
Form8.cs	Екранна форма пошуку та перегляду документу «Облік відпрацьованого часу»
Form9.cs	Екранна форма пошуку, перегляду та графічного відображення документу «Облік відпрацьованого часу»

Екранні форми, які відображають нормативно-довідникову інформацію, представлені на рисунках 3.8 – 3.11. Форма, яка відображає вихідну інформацію, зображена на рисунку 3.12. Для виконання операцій додавання, оновлення, видалення та пошуку даних у складових частинах довідникової або вхідної інформації користувачам необхідно обрати відповідний пункт з головного меню системи.

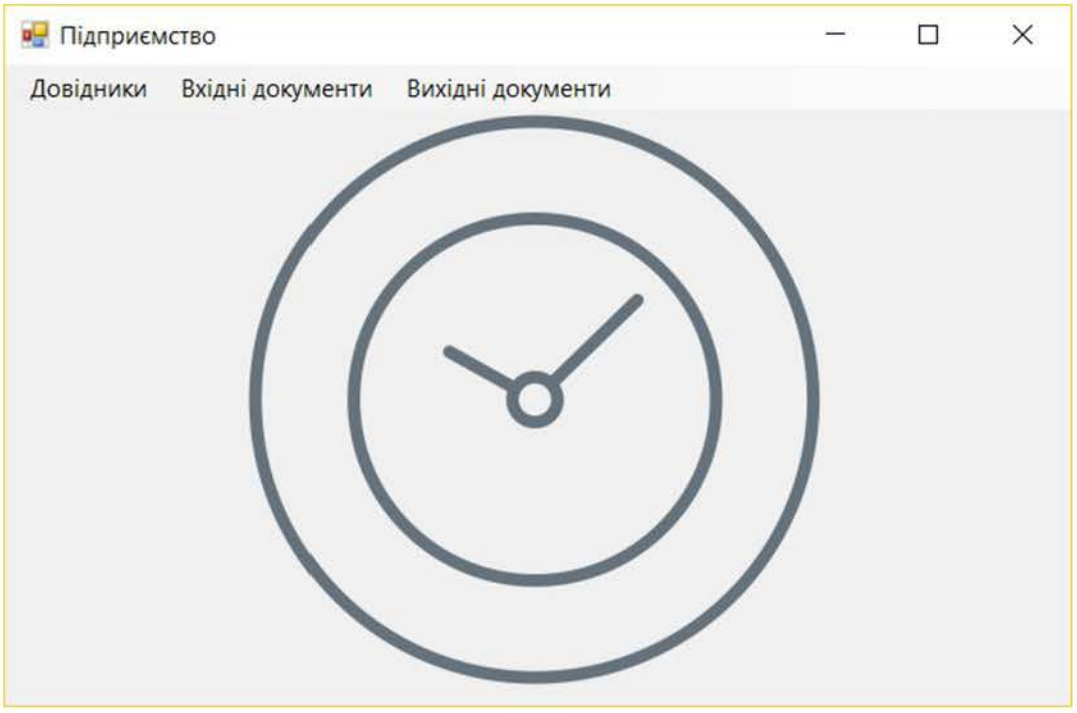


Рисунок 3.8 – Головна екранна форма

Екранні форми для довідників співробітників, позначок, відділів та посад відображені на рисунках 3.8 – 3.11. Процес взаємодії користувача з системою має однакову схему на кожній екранній формі і включає такі дії:

Для додавання нового запису до бази даних необхідно ввести в текстові поля на екранних формах відповідну інформацію і натиснути кнопку «Додати» або «Зберегти». Після цього внесені дані будуть відображені в табличній частині форми.

Для видалення запису з бази даних користувачу потрібно вибрати в таблиці відповідний запис і натиснути кнопку «Видалити». Після цього користувач отримає повідомлення про підтвердження операції видалення.

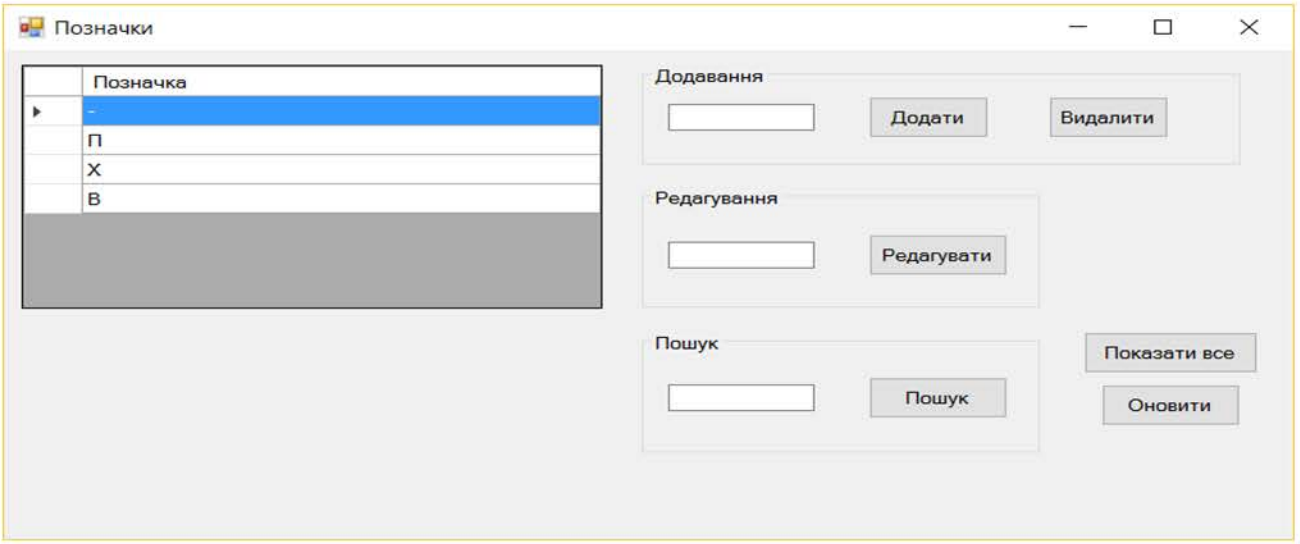


Рисунок 3.9 – Екранна форма довідника «Позначки»

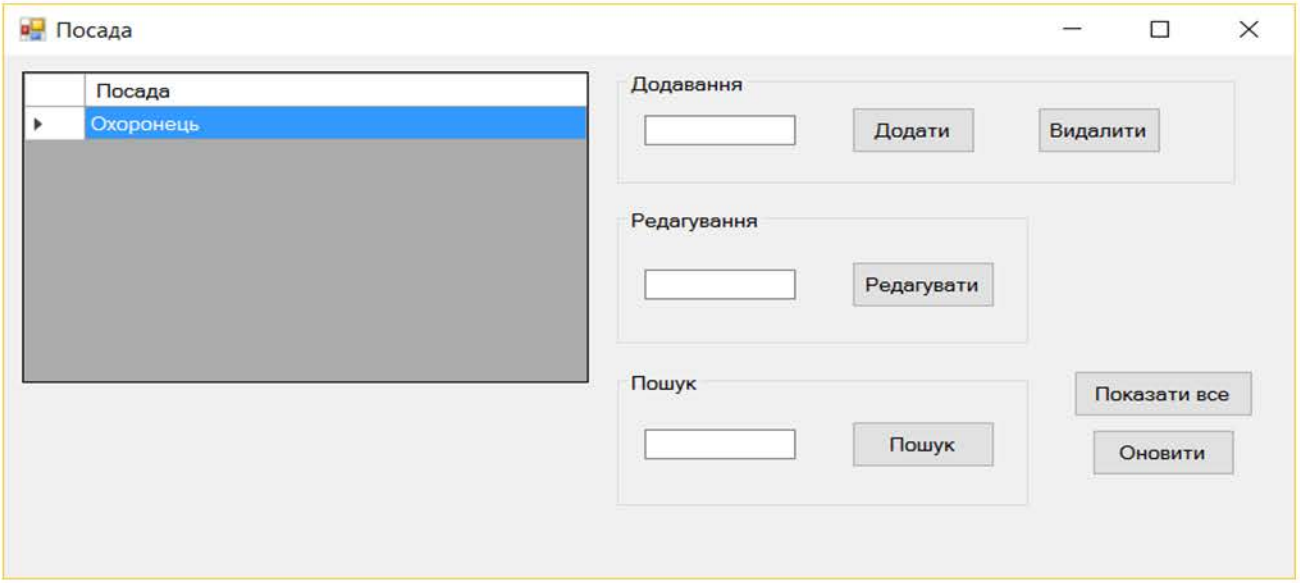


Рисунок 3.10 – Екранна форма довідника «Посада»

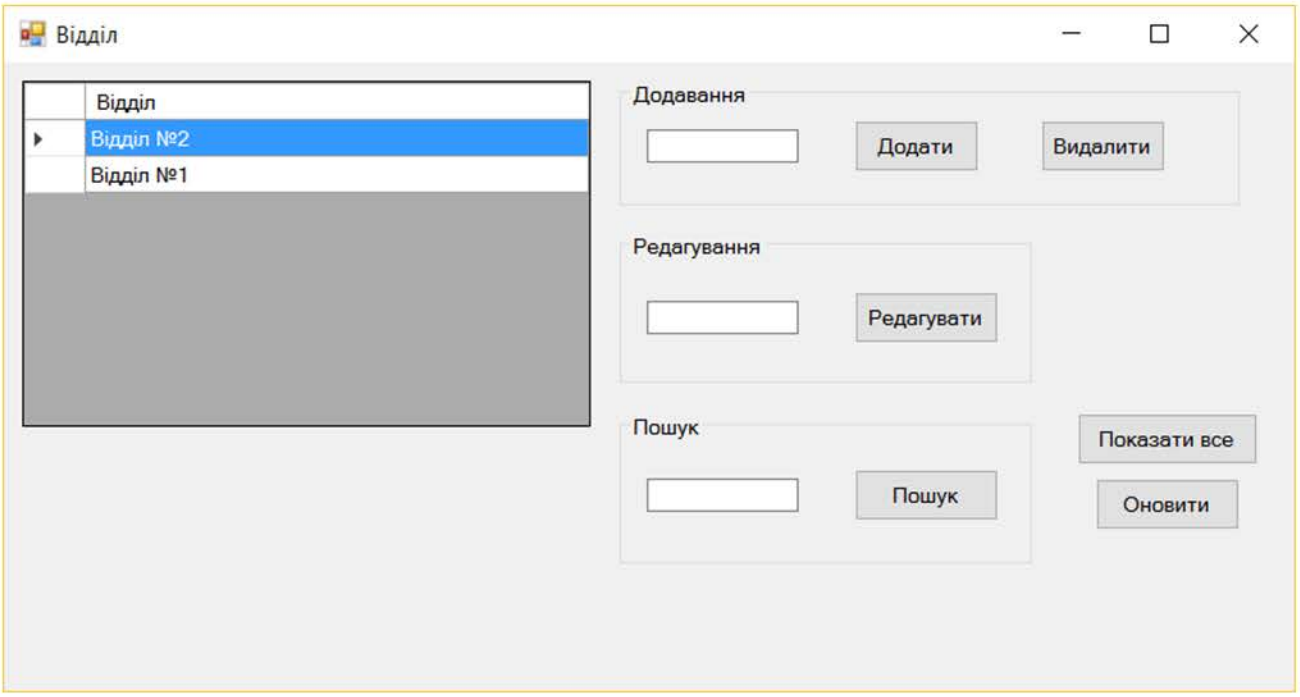


Рисунок 3.11 – Екранна форма довідника «Відділ»

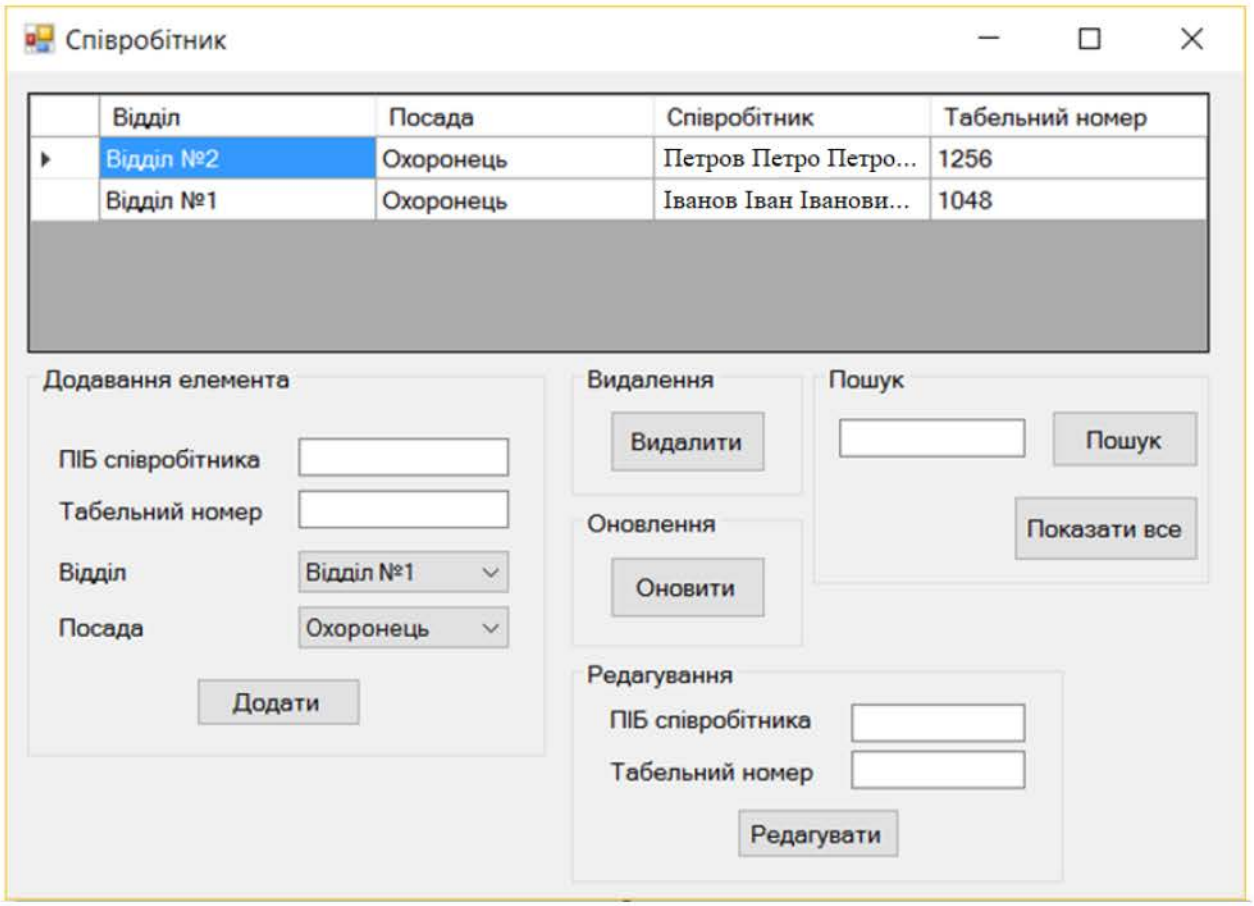


Рисунок 3.12 – Екранна форма довідника «Співробітник»

Для внесення змін до запису, що зберігається в базі даних, користувачу необхідно відредагувати відповідне текстове поле та натиснути кнопку «Редагувати».

Для пошуку конкретного запису потрібно встановити відповідний параметр запиту до бази даних та натиснути кнопку «Пошук». Після цього результат пошуку буде відображено в таблиці.

Екранні форми вхідної інформації системи розроблені з урахуванням принципів уніфікації та стандартизації. Діалог між користувачем та системою має однакову структуру на кожній екранній формі. У випадку форми перегляду документу процес взаємодії користувача з системою виглядає наступним чином:

Для видалення таблиці відпрацьованого часу користувачу потрібно обрати відповідний табель у відповідній табличній частині екранної форми та натиснути кнопку «Видалити».

Для здійснення пошуку табелю за датою його створення серед всіх записів у базі даних, користувачу необхідно вибрати потрібний діапазон дат та натиснути кнопку «Пошук».

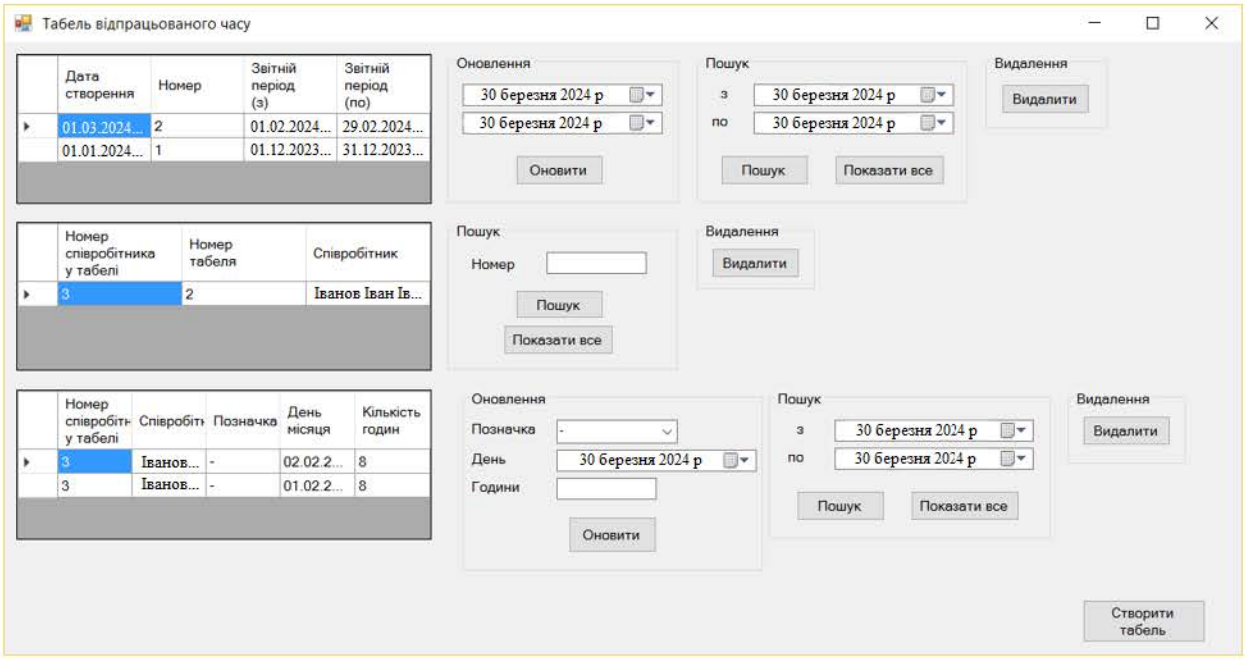


Рисунок 3.13 – Екранна форма документу «Табель відпрацьованого часу»

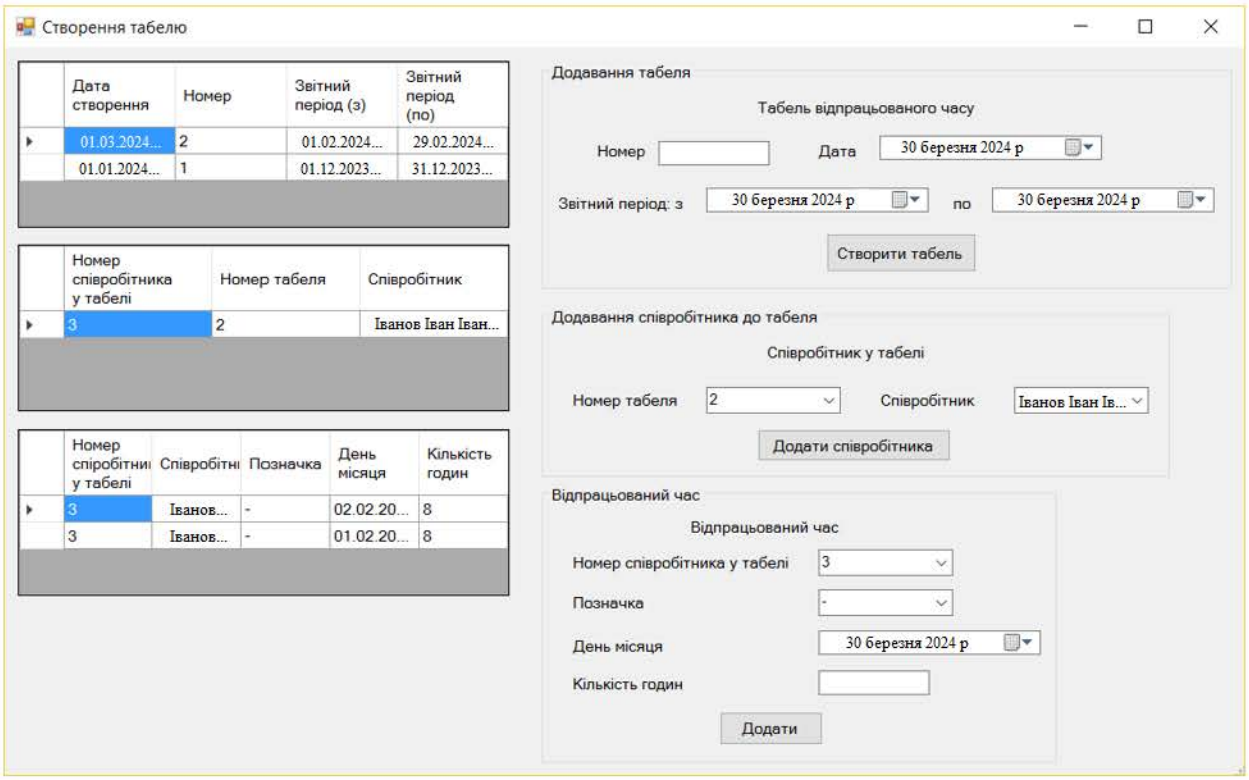


Рисунок 3.14 – Екранна форма «Додавання табелю відпрацьованого часу»

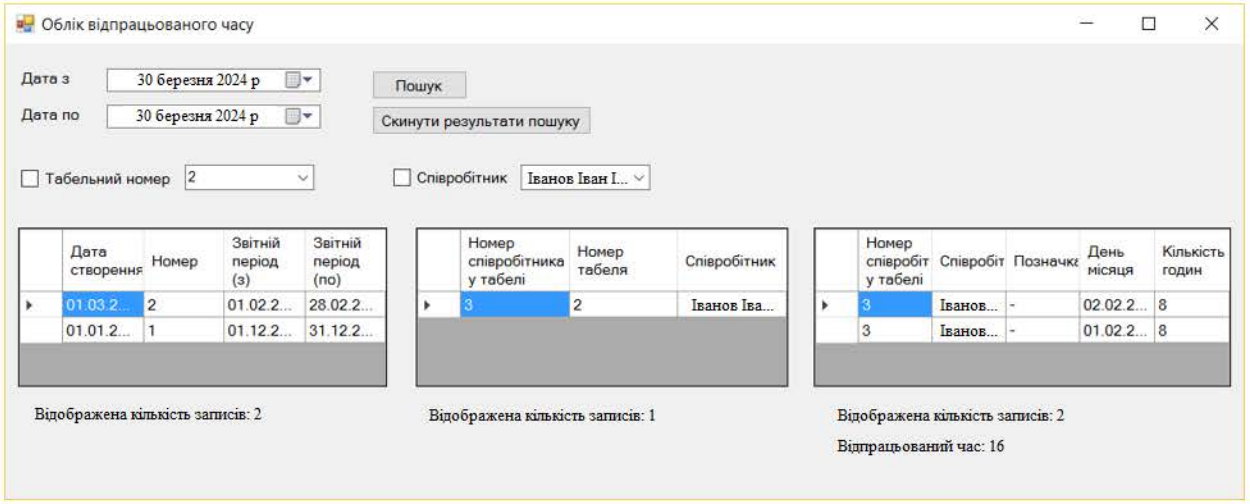


Рисунок 3.15 – Екранна форма документу «Облік відпрацьованого часу»

3.8 Висновки до третього розділу

У розділі реалізації програмного забезпечення було проведено

детальний аналіз та розроблено систему обліку робочого часу на основі мови програмування C#, бази даних MSSQLServer та інших технологій.

Мова програмування C#, яка є однією з найпопулярніших у сфері розробки програмного забезпечення, була обрана для написання основної логіки програми. Використання бази даних MSSQLServer дозволило створити надійне сховище для зберігання та обробки даних про робочий час працівників. Крім того, в реалізації були використані інші компоненти, такі як ADO.NET для забезпечення доступу до бази даних та різноманітні бібліотеки для реалізації функціональності інтерфейсу користувача.

Процес розробки включав в себе створення функціональних моделей, розробку інтерфейсу користувача, імплементацію алгоритмів обробки даних та тестування рішення на реальних даних.

Результатом роботи є функціональна система, яка дозволяє ефективно вести облік робочого часу працівників, забезпечуючи точність, швидкість та зручність використання. Впровадження цієї системи може значно підвищити продуктивність робочих процесів та оптимізувати використання робочого часу персоналу.

ВИСНОВКИ

Мета роботи полягає у розробці програмного забезпечення обліку відпрацьованого часу співробітників з можливістю налаштування та персоналізації для підвищення ефективності управління персоналом на підприємстві.

Розроблене програмного забезпечення вирішує наступні задачі:

1) Автоматизація процесу обліку робочого часу:

– забезпечити точний та автоматизований облік робочого часу співробітників для зменшення ручного введення даних та зниження ризику помилок.

2) Підвищення ефективності управління персоналом:

– надати керівництву інструменти для ефективного управління робочим часом, відпустками та відсутностями співробітників.

– сприяти прийняттю обґрунтованих управлінських рішень на основі точних даних та аналітики.

3) Забезпечення гнучкості та індивідуалізації:

– дозволяє налаштовувати графіки роботи, права доступу та індивідуальні налаштування користувачів для відповідності потребам різних відділів та співробітників.

4) Інтеграція з іншими інформаційними системами:

5) Підвищення прозорості та контрольованості процесів:

– надати інструменти для прозорого відстеження робочого часу та активності співробітників, що дозволяє керівництву ефективніше контролювати процеси та оперативно реагувати на відхилення.

6) Оптимізація витрат на оплату праці:

– забезпечити точний облік відпрацьованого часу для оптимізації витрат на оплату праці та уникнення помилок у нарахуванні зарплати.

Об'єкт дослідження:

Об’єктом дослідження є процеси управління персоналом на

підприємстві, зокрема облік робочого часу співробітників. Це включає всі аспекти реєстрації, моніторингу та аналізу відпрацьованого часу, а також інтеграцію цих процесів у загальну систему управління підприємством.

Предметом дослідження є програмне забезпечення обліку робочого часу співробітників. Це включає її архітектуру, функціональні можливості, технології реалізації та ефективність у контексті управління персоналом. Особлива увага приділяється аналізу програмного забезпечення, яке використовується для реалізації системи обліку часу, таких як C#, Microsoft SQL Server та ADO.NET.

Таким чином, робота досліджує об'єкт в контексті управління персоналом і аналізує конкретні технології та методи, які використовуються для автоматизації обліку робочого часу як предмет дослідження.

Актуальність розробки та впровадження системи обліку робочого часу співробітників на підприємстві обумовлена кількома ключовими факторами:

1) Підвищення ефективності управління персоналом:

– У сучасному бізнес-середовищі, де конкурентоспроможність визначається ефективністю використання ресурсів, автоматизація обліку робочого часу стає необхідністю. Впровадження такої системи дозволяє значно зменшити адміністративні витрати, оптимізувати робочі процеси та підвищити продуктивність праці.

2) Точність та прозорість даних

– Ручний облік робочого часу часто призводить до помилок і неточностей, що може викликати конфлікти між співробітниками та керівництвом. Автоматизована система забезпечує високу точність даних та прозорість процесів, що сприяє підвищенню довіри між працівниками та адміністрацією.

3) Дотримання законодавства:

– Сучасні трудові закони вимагають точного обліку робочого часу для забезпечення прав працівників. Система обліку робочого часу допомагає підприємству відповідати цим вимогам, знижуючи ризик юридичних проблем.

4) Аналіз та звітність:

– Здатність автоматично генерувати звіти і аналізувати дані дозволяє керівництву приймати обґрунтовані рішення, що сприяє покращенню стратегічного планування і управління людськими ресурсами.

5) Інтеграція з іншими системами:

– Впровадження системи обліку робочого часу дозволяє інтегрувати її з іншими інформаційними системами підприємства, створюючи єдиний інформаційний простір. Це спрощує процеси управління та підвищує загальну ефективність підприємства.

6) Покращення планування та прогнозування:

– Зібрані дані дозволяють більш точно планувати робочі графіки, прогнозувати потреби у персоналі та розробляти ефективні стратегії управління людськими ресурсами, що сприяє зменшенню витрат і підвищенню продуктивності.

Враховуючи вищезазначене, розробка програмного забезпечення обліку робочого часу співробітників є надзвичайно актуальним завданням для будь-якого сучасного підприємства, яке прагне досягти високих результатів та підтримувати конкурентоспроможність на ринку.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 – «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

- Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

- Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

- Здатність розробляти архітектури, модулі та компоненти програмних систем;

- Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним

завданням та стандартами;

- Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу;

- Володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

- Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

- Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

- Здатність до алгоритмічного та логічного мислення тощо.

Серед результатів навчання, визначених стандартом, кваліфікаційна робота реалізує наступні:

- Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

- Вміти розробляти людино-машинний інтерфейс;

- Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

- Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

- Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

- Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення;

- Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних;

- Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення;

- Знати та вміти застосовувати інформаційні технології обробки,

зберігання та передачі даних;

- Вміти документувати та презентувати результати розробки програмного забезпечення.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

– здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

– здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

– здатність розробляти архітектури, модулі та компоненти програмних систем;

– володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

– здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

– здатність до алгоритмічного та логічного мислення.

Серед програмних результатів, визначених стандартом, кваліфікаційна робота реалізовує наступні:

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

– сміти розробляти людино-машинний інтерфейс;

– знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

– проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

– вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

– мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

– знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

– вміти документувати та презентувати результати розробки програмного забезпечення тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Joseph Albahari, Ben Albahari. *C# 9.0 in a Nutshell: The Definitive Reference*, 2021.
2. Andrew Troelsen, Philip Japikse. *Pro C# 8 with .NET Core: Foundational Principles and Practices*, 2019.
3. Mark J. Price. *C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development*, 2019.
4. Christian Nagel. *Professional C# and .NET: 2021 Edition*, 2021.
5. Joseph Albahari, Ben Albahari. *C# 8.0 Pocket Reference: Instant Help for C# 8.0 Programmers*, 2019.
6. Itzik Ben-Gan. *T-SQL Querying*, 2019.
7. Kalen Delaney, Jonathan Kehayias, Erin Stellato. *SQL Server 2019 Internals*, 2020.
8. Dusan Petkovic. *Microsoft SQL Server 2019: A Beginner's Guide, Seventh Edition*, 2020.
9. Benjamin Nevarez. *High Performance SQL Server: The Go Faster Book*, 2020.
10. William Durkin, Milos Radivojevic, Dejan Sarka. *SQL Server 2019 Administration Inside Out*, 2020.
11. Gregory A. Beamer, Kevin S. Goff. *Professional ADO.NET 2: Programming with SQL Server, Oracle & MySQL*, 2019.
12. Sayed Y. Hashimi, Erik Rank. *Pro ADO.NET Data Services: Working with RESTful Data*, 2019.
13. John Paul Mueller, Bill Hamilton. *ADO.NET 3.5 Cookbook: Building Data-Centric .NET Applications*, 2019.
14. Glenn Johnson. *Programming Microsoft ADO.NET 4: Essential Skills for Database Professionals*, 2020.
15. Roger Jennings. *Professional ADO.NET 3.5 with LINQ and the Entity Framework*, 2020.

ДОДАТОК А

Лістинг створення бази даних

```
create database db_pidpruemstvo
use db_pidpruemstvo

create table Posada
(
	ID_Posada int primary key identity(1,1) not null,
	Posada_Name varchar(100)
)

create table Viddil
(
	ID_Viddil int primary key identity(1,1) not null,
	Viddil_Name varchar(50)
)

create table Mark
(
	ID_Mark int primary key identity(1,1) not null,
	Mark_Name varchar(10)
)

create table Spivrobitnuk
(
	ID_Spivrobitnuk int primary key identity(1,1) not null,
	ID_Viddil int foreign key REFERENCES Viddil(ID_Viddil),
	ID_Posada int foreign key REFERENCES Posada(ID_Posada),
	Spivrobitnuk_Name varchar(50),
	Tabel_Number int
)

create table Tabel_vc
(
	ID_Tabel_vc int primary key identity(1,1) not null,
	Date_create datetime,
	Number int,
	Zvit_period_z datetime,
	Zvit_period_po datetime
)
```

```
create table SpivrobitnukTabel
(
	ID_SpivrobitnukTabel int primary key identity(1,1) not null,
	ID_Tabel_vc int foreign key REFERENCES Tabel_vc(ID_Tabel_vc),
	ID_Spivrobitnuk int foreign key REFERENCES
Spivrobitnuk(ID_Spivrobitnuk)
)

create table VidpracTime
(
	ID_VidpracTime int primary key identity(1,1) not null,
	ID_SpivrobitnukTabel int foreign key REFERENCES
SpivrobitnukTabel(ID_SpivrobitnukTabel),
	ID_Mark int foreign key REFERENCES Mark(ID_Mark),
	Den_mis datetime,
	Kilk_god int
)
```

ДОДАТОК Б

Лістинг створення програмного забезпечення

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void позначкиToolStripMenuItem_Click(object sender, EventArgs e)
      {
         Form2 form2 = new Form2();
         form2.Show();
      }

      private void посадаToolStripMenuItem_Click(object sender, EventArgs e)
      {
         Form3 form3 = new Form3();
         form3.Show();
      }

      private void відділToolStripMenuItem_Click(object sender, EventArgs e)
      {
         Form4 form4 = new Form4();
         form4.Show();
      }

      private void співробітникToolStripMenuItem_Click(object sender,
```

```
EventArgs e)
        {
            Form5 form5 = new Form5();
            form5.Show();
        }

        private void табельВідпрацьованогоЧасуToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Form6 form6 = new Form6();
            form6.Show();
        }

        private void обліkВідпрацьованогоЧасуToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Form8 form8 = new Form8();
            form8.Show();
        }

        private void графічнеПредставленняToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Form9 form9 = new Form9();
            form9.Show();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
```

```
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                MessageBox.Show("Заповніть поле додавання");
                textBox1.Focus();
            }
            else
            {
                sqlConnection1.Open();
                dataSet11.Clear();
                sqlDataAdapter1.InsertCommand.Parameters[0].Value = textBox1.Text;
                sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
                sqlConnection1.Close();
                sqlDataAdapter1.Fill(dataSet11);
                textBox1.Text = "";
            }
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
            sqlDataAdapter1.Fill(dataSet11);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                MessageBox.Show("Видалити неможливо");
            }
            else
            {
                if (MessageBox.Show("Ви дійсно хочете видалити цю позначку?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
```

```
            {
                sqlDataAdapter1.DeleteCommand.Parameters[0].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                sqlConnection1.Open();
                sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
                sqlConnection1.Close();
                dataSet11.Mark.Clear();
                sqlDataAdapter1.Fill(dataSet11);
            }
        }
    }

    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '-'))
        {
            e.Handled = true;
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (textBox2.Text == "")
        {
            MessageBox.Show("Заповнять поле для редагування");
        }
        else
        {
            sqlDataAdapter1.UpdateCommand.Parameters[1].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
            sqlDataAdapter1.UpdateCommand.Parameters[2].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
            sqlDataAdapter1.UpdateCommand.Parameters[0].Value =
textBox2.Text;
            sqlConnection1.Open();
            sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            dataSet11.Mark.Clear();
            sqlDataAdapter1.Fill(dataSet11);
            MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            textBox2.Text = "";
        }
```

```
        }

        private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '-'))
            {
                e.Handled = true;
            }
        }

        private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '-'))
            {
                e.Handled = true;
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
            sqlDataAdapter1.Fill(dataSet11);
        }

        private void button6_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.Update(dataSet11);
            MessageBox.Show("Дані успішно оновлені");
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox2.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%" +
textBox3.Text + "%";
            dataSet11.Clear();
            sqlDataAdapter1.Fill(dataSet11);
```

```
            if (this.BindingContext[dataSet11, "Mark"].Count == 0)
            {
                MessageBox.Show("Таких даних не існує");
                textBox3.Focus();
            }
            else if (textBox3.Text == "")
            {
                MessageBox.Show("Заповніть поле пошуку!");
                textBox3.Focus();
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
            sqlDataAdapter1.Fill(dataSet21);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                MessageBox.Show("Заповніть поле додавання");
                textBox1.Focus();
```

```
        }
        else
        {
            sqlConnection1.Open();
            dataSet21.Clear();
            sqlDataAdapter1.InsertCommand.Parameters[0].Value = textBox1.Text;
            sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            sqlDataAdapter1.Fill(dataSet21);
            textBox1.Text = "";
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (dataGridView1.Rows.Count == 0)
        {
            MessageBox.Show("Видалити неможливо");
        }
        else
        {
            if (MessageBox.Show("Ви дійсно хочете видалити цю посаду?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {
                sqlDataAdapter1.DeleteCommand.Parameters[0].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                sqlConnection1.Open();
                sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
                sqlConnection1.Close();
                dataSet21.Posada.Clear();
                sqlDataAdapter1.Fill(dataSet21);
            }
        }
    }

    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((!Char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
        {
            e.Handled = true;
        }
    }
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            if (textBox2.Text == "")
            {
                MessageBox.Show("Заповнять поле для редагування");
            }
            else
            {
                sqlDataAdapter1.UpdateCommand.Parameters[1].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                sqlDataAdapter1.UpdateCommand.Parameters[2].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                sqlDataAdapter1.UpdateCommand.Parameters[0].Value =
textBox2.Text;
                sqlConnection1.Open();
                sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
                sqlConnection1.Close();
                dataSet21.Posada.Clear();
                sqlDataAdapter1.Fill(dataSet21);
                MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                textBox2.Text = "";
            }
        }

        private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((!Char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%" +
textBox3.Text + "%";
            dataSet21.Clear();
            sqlDataAdapter1.Fill(dataSet21);
            if (this.BindingContext[dataSet21, "Posada"].Count == 0)
            {
                MessageBox.Show("Таких даних не існує");
                textBox3.Focus();
            }
```

```
            else if (textBox3.Text == "")
            {
                MessageBox.Show("Заповніть поле пошуку!");
                textBox3.Focus();
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
            sqlDataAdapter1.Fill(dataSet21);
        }

        private void button6_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.Update(dataSet21);
            MessageBox.Show("Дані успішно оновлені");
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox2.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }

        private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
        {
            if ((!Char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace Підприємство
{
   public partial class Form4 : Form
   {
      public Form4()
      {
         InitializeComponent();
      }

      private void Form4_Load(object sender, EventArgs e)
      {
         sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
         sqlDataAdapter1.Fill(dataSet31);
      }

      private void button1_Click(object sender, EventArgs e)
      {
         if (textBox1.Text == "")
         {
            MessageBox.Show("Заповніть поле додавання");
            textBox1.Focus();
         }
         else
         {
            sqlConnection1.Open();
            dataSet31.Clear();
            sqlDataAdapter1.InsertCommand.Parameters[0].Value = textBox1.Text;
            sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            sqlDataAdapter1.Fill(dataSet31);
            textBox1.Text = "";
         }
      }

      private void button2_Click(object sender, EventArgs e)
      {
         if (dataGridView1.Rows.Count == 0)
         {
            MessageBox.Show("Видалити неможливо");
         }
         else
         {
            if (MessageBox.Show("Ви дійсно хочете видалити цю посаду?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
```

```
DialogResult.Yes)
            {
                sqlDataAdapter1.DeleteCommand.Parameters[0].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                sqlConnection1.Open();
                sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
                sqlConnection1.Close();
                dataSet31.Viddil.Clear();
                sqlDataAdapter1.Fill(dataSet31);
            }
        }
    }

    private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '№') && (e.KeyChar != ' '))
        {
            e.Handled = true;
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (textBox2.Text == "")
        {
            MessageBox.Show("Заповнять поле для редагування");
        }
        else
        {
            sqlDataAdapter1.UpdateCommand.Parameters[1].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
            sqlDataAdapter1.UpdateCommand.Parameters[2].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
            sqlDataAdapter1.UpdateCommand.Parameters[0].Value =
textBox2.Text;
            sqlConnection1.Open();
            sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            dataSet31.Viddil.Clear();
            sqlDataAdapter1.Fill(dataSet31);
            MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            textBox2.Text = "";
```

```
        }
    }

    private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '№') && (e.KeyChar != ' '))
        {
            e.Handled = true;
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%" +
textBox3.Text + "%";
        dataSet31.Clear();
        sqlDataAdapter1.Fill(dataSet31);
        if (this.BindingContext[dataSet31, "Viddil"].Count == 0)
        {
            MessageBox.Show("Таких даних не існує");
            textBox3.Focus();
        }
        else if (textBox3.Text == "")
        {
            MessageBox.Show("Заповніть поле пошуку!");
            textBox3.Focus();
        }
    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (!Char.IsLetter(e.KeyChar)) &&
(e.KeyChar != (char)Keys.Back) && (e.KeyChar != '№') && (e.KeyChar != ' '))
        {
            e.Handled = true;
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
        sqlDataAdapter1.Fill(dataSet31);
    }
```

```
        private void button6_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.Update(dataSet31);
            MessageBox.Show("Дані успішно оновлені");
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            textBox2.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Підприємство
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
            sqlDataAdapter1.Fill(dataSet41.Spivrobitnuk);

            SqlConnection con = new SqlConnection("Data
Source=CBETA\\SQLEXPRESS;Initial Catalog=db_pidpruemstvo;Integrated
Security=True");

            con.Open();
```

```
            string strCmd = "SELECT * from Viddil WHERE Viddil_Name NOT
LIKE 'Видалено'";
            string strCmd1 = "SELECT * from Posada WHERE Posada_Name NOT
LIKE 'Видалено'";
            SqlCommand cmd = new SqlCommand(strCmd, con);
            SqlCommand cmd1 = new SqlCommand(strCmd1, con);
            SqlDataAdapter da = new SqlDataAdapter(strCmd, con);
            SqlDataAdapter da1 = new SqlDataAdapter(strCmd1, con);
            DataSet ds = new DataSet();
            DataSet ds1 = new DataSet();
            da.Fill(ds);
            da1.Fill(ds1);
            cmd.ExecuteNonQuery();
            cmd1.ExecuteNonQuery();
            con.Close();

            comboBox1.DisplayMember = "Viddil_Name";
            comboBox1.ValueMember = "ID_Viddil";
            comboBox1.DataSource = ds.Tables[0];

            comboBox2.DisplayMember = "Posada_Name";
            comboBox2.ValueMember = "ID_Posada";
            comboBox2.DataSource = ds1.Tables[0];
        }

        private void button1_Click(object sender, EventArgs e)
        {
            sqlConnection1.Open();
            dataSet41.Clear();
            sqlDataAdapter1.InsertCommand.Parameters[0].Value =
comboBox1.SelectedValue;
            sqlDataAdapter1.InsertCommand.Parameters[1].Value =
comboBox2.SelectedValue;
            sqlDataAdapter1.InsertCommand.Parameters[2].Value = textBox1.Text;
            sqlDataAdapter1.InsertCommand.Parameters[3].Value =
Convert.ToInt32(textBox2.Text);
            sqlDataAdapter1.InsertCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            sqlDataAdapter1.Fill(dataSet41);
            textBox1.Clear();
            textBox2.Clear();
        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
```

```
        {
            if ((!Char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != ' '))
            {
                e.Handled = true;
            }
        }

        private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
            {
                e.Handled = true;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                MessageBox.Show("Видалити неможливо");
            }
            else
            {
                if (MessageBox.Show("Ви дійсно хочете видалити цього
співробітника?", "Підтвердження", MessageBoxButtons.YesNo,
MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    sqlDataAdapter1.DeleteCommand.Parameters[0].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
                    sqlConnection1.Open();
                    sqlDataAdapter1.DeleteCommand.ExecuteNonQuery();
                    sqlConnection1.Close();
                    dataSet41.Spivrobitnuk.Clear();
                    sqlDataAdapter1.Fill(dataSet41);
                    MessageBox.Show("Видалено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.Update(dataSet41);
```

```
        MessageBox.Show("Запис збережено до БД", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void button4_Click(object sender, EventArgs e)
    {
        sqlDataAdapter1.SelectCommand.Parameters["@Param1"].Value = "%" +
textBox3.Text + "%";
        dataSet41.Clear();
        sqlDataAdapter1.Fill(dataSet41.Spivrobitnuk);
        if (this.BindingContext[dataSet41, "Spivrobitnuk"].Count == 0)
        {
            MessageBox.Show("Нічого не знайдено!");
            textBox3.Focus();
            textBox3.Text = "";
        }
        else if (textBox3.Text == "")
        {
            MessageBox.Show("Заповніть поле пошуку!");
            textBox3.Focus();
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        sqlDataAdapter1.SelectCommand.Parameters[0].Value = "%%";
        sqlDataAdapter1.Fill(dataSet41);
    }

    private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
    {

    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (textBox4.Text == "" && textBox5.Text == "")
        {
            MessageBox.Show("Заповніть поля для редагування");
        }
        else
        {
            sqlDataAdapter1.UpdateCommand.Parameters[2].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
```

```
            sqlDataAdapter1.UpdateCommand.Parameters[3].Value =
dataGridView1.CurrentRow.Cells[0].Value.ToString();
            sqlDataAdapter1.UpdateCommand.Parameters[0].Value =
textBox4.Text;
            sqlDataAdapter1.UpdateCommand.Parameters[1].Value =
textBox5.Text;
            sqlConnection1.Open();
            sqlDataAdapter1.UpdateCommand.ExecuteNonQuery();
            sqlConnection1.Close();
            dataSet41.Spivrobitnuk.Clear();
            sqlDataAdapter1.Fill(dataSet41);
            MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            textBox4.Text = "";
            textBox5.Text = "";
        }
    }

    private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        textBox4.Text = "";
        textBox5.Text = "";
        textBox4.Text = dataGridView1.CurrentRow.Cells[5].Value.ToString();
        textBox5.Text = dataGridView1.CurrentRow.Cells[6].Value.ToString();
    }

    private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
    {
        if ((!Char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back) &&
(e.KeyChar != ' '))
        {
            e.Handled = true;
        }
    }

    private void textBox5_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && (e.KeyChar != (char)Keys.Back))
        {
            e.Handled = true;
        }
    }
```

```
        private void dataSet41BindingSource_CurrentChanged(object sender,
EventArgs e)
        {

        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();
            button1.Click += (sender, e) =>
            {
                Form7 f7 = new Form7();
                f7.FormClosed += (obj, arg) =>
                {
                    tabelDataSet.Clear();

this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
                    this.tabel_vcTableAdapter.FillBy(this.tabelDataSet.Tabel_vc);
                };
                f7.ShowDialog();
            };
        }

        private void Form6_Load(object sender, EventArgs e)
        {
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"markDataSet.Mark". При необхідності вона може бути переміщена або
```

```
видалена.
            this.markTableAdapter.Fill(this.markDataSet.Mark);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.VidpracTime". При необхідності вона може бути переміщена
або видалена.
            this.vidpracTimeTableAdapter.Fill(this.tabelDataSet.VidpracTime);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути
переміщена або видалена.

this.spivrobitnukTabelTableAdapter.Fill(this.tabelDataSet.SpivrobitnukTabel);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або
видалена.
            this.tabel_vcTableAdapter.Fill(this.tabelDataSet.Tabel_vc);

            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"markDataSet.Mark". При необхідності вона може бути переміщена або
видалена.
            this.markTableAdapter.FillBy(this.markDataSet.Mark);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.VidpracTime". При необхідності вона може бути переміщена
або видалена.
            this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути
переміщена або видалена.

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або
видалена.
            this.tabel_vcTableAdapter.FillBy(this.tabelDataSet.Tabel_vc);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //Form7 form7 = new Form7();
            //form7.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            tabel_vcTableAdapter.UpdateQuery(dateTimePicker1.Value,
```

```
dateTimePicker2.Value, (int)dataGridView1.CurrentRow.Cells[0].Value,
(int)dataGridView1.CurrentRow.Cells[0].Value);
            MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            tabel_vcTableAdapter.Update(tabelDataSet.Tabel_vc);
            Refresh();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            dateTimePicker1.DataBindings.Add(new Binding("Text",
tabelvcBindingSource, "Zvit_period_z", true));
            dateTimePicker2.DataBindings.Add(new Binding("Text",
tabelvcBindingSource, "Zvit_period_po", true));
        }

        private void button3_Click(object sender, EventArgs e)
        {

vidpracTimeTableAdapter.UpdateQuery2(Convert.ToInt32(comboBox1.SelectedV
alue.ToString()), dateTimePicker3.Value, Convert.ToInt32(textBox1.Text),
(int)dataGridView3.CurrentRow.Cells[0].Value,
(int)dataGridView3.CurrentRow.Cells[0].Value);
            MessageBox.Show("Оновлено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            vidpracTimeTableAdapter.Update(tabelDataSet.VidpracTime);
            Refresh();
        }

        private void dataGridView3_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            comboBox1.DataBindings.Add(new Binding("Text",
fKVidpracTiIDSp0CBAE877BindingSource, "Mark_Name", true));
            dateTimePicker3.DataBindings.Add(new Binding("Text",
fKVidpracTiIDSp0CBAE877BindingSource, "Den_mis", true));
            textBox1.DataBindings.Add(new Binding("Text",
fKVidpracTiIDSp0CBAE877BindingSource, "Kilk_god", true));
        }

        private void button4_Click(object sender, EventArgs e)
        {
            tabel_vcTableAdapter.FillByDate(tabelDataSet.Tabel_vc,
```

```
Convert.ToDateTime(dateTimePicker4.Value.ToShortDateString()),
Convert.ToDateTime(dateTimePicker5.Value.ToShortDateString()));
        }

        private void button5_Click(object sender, EventArgs e)
        {
            this.tabel_vcTableAdapter.FillBy(this.tabelDataSet.Tabel_vc);

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
            this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
        }

        private void button7_Click(object sender, EventArgs e)
        {
            vidpracTimeTableAdapter.FillByDate2(tabelDataSet.VidpracTime,
Convert.ToDateTime(dateTimePicker6.Value.ToShortDateString()),
Convert.ToDateTime(dateTimePicker7.Value.ToShortDateString()));
        }

        private void button6_Click(object sender, EventArgs e)
        {
            this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
        }

        private void button8_Click(object sender, EventArgs e)
        {

spivrobitnukTabelTableAdapter.FillByName(tabelDataSet.SpivrobitnukTabel,
"%"+textBox2.Text+"%");
        }

        private void button9_Click(object sender, EventArgs e)
        {

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
            this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
        }

        private void button10_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                MessageBox.Show("Видалити неможливо");
            }
```

```
        else
        {
            if (MessageBox.Show("Ви дійсно бажаєте видалити?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {

tabel_vcTableAdapter.DeleteQuery((int)dataGridView1.CurrentRow.Cells[0].Valu
e);
                tabel_vcTableAdapter.Update(tabelDataSet.Tabel_vc);
                tabel_vcTableAdapter.FillBy(tabelDataSet.Tabel_vc);
                MessageBox.Show("Видалено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
    }

    private void button11_Click(object sender, EventArgs e)
    {
        if (dataGridView2.Rows.Count == 0)
        {
            MessageBox.Show("Видалити неможливо");
        }
        else
        {
            if (MessageBox.Show("Ви дійсно бажаєте видалити?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
            {

spivrobitnukTabelTableAdapter.DeleteQuery((int)dataGridView2.CurrentRow.Cel
ls["iDSpivrobitnukTabelDataGridViewTextBoxColumn"].Value);

spivrobitnukTabelTableAdapter.Update(tabelDataSet.SpivrobitnukTabel);

spivrobitnukTabelTableAdapter.FillBy(tabelDataSet.SpivrobitnukTabel);
                MessageBox.Show("Видалено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
    }

    private void button12_Click(object sender, EventArgs e)
    {
```

```
            if (dataGridView3.Rows.Count == 0)
            {
                MessageBox.Show("Видалити неможливо");
            }
            else
            {
                if (MessageBox.Show("Ви дійсно бажаєте видалити?",
"Підтвердження", MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
                {

vidpracTimeTableAdapter.DeleteQuery((int)dataGridView3.CurrentRow.Cells["i
DVidpracTimeDataGridViewTextBoxColumn"].Value);
                    vidpracTimeTableAdapter.Update(tabelDataSet.VidpracTime);
                    vidpracTimeTableAdapter.FillBy(tabelDataSet.VidpracTime);
                    MessageBox.Show("Видалено успішно", "Результат",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
    public partial class Form7 : Form
    {
        public Form7()
        {
            InitializeComponent();
        }

        private void Form7_Load(object sender, EventArgs e)
        {
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
```

"tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути переміщена або видалена.

this.spivrobitnukTabelTableAdapter.Fill(this.tabelDataSet.SpivrobitnukTabel);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "markDataSet.Mark". При необхідності вона може бути переміщена або видалена.
this.markTableAdapter.Fill(this.markDataSet.Mark);

// TODO: дана строка коду дозволяє завантажити дані до таблиці "spivrobDataSet.Spivrobitnuk". При необхідності вона може бути переміщена або видалена.
this.spivrobitnukTableAdapter.Fill(this.spivrobDataSet.Spivrobitnuk);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.VidpracTime". При необхідності вона може бути переміщена або видалена.
this.vidpracTimeTableAdapter.Fill(this.tabelDataSet.VidpracTime);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або видалена.
this.tabel_vcTableAdapter.Fill(this.tabelDataSet.Tabel_vc);

// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути переміщена або видалена.

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "markDataSet.Mark". При необхідності вона може бути переміщена або видалена.
this.markTableAdapter.FillBy(this.markDataSet.Mark);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.VidpracTime". При необхідності вона може бути переміщена або видалена.
this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути переміщена або видалена.

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
// TODO: дана строка коду дозволяє завантажити дані до таблиці "tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або видалена.
this.tabel_vcTableAdapter.FillBy(this.tabelDataSet.Tabel_vc);

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (textBox1.Text.Length > 0 && dateTimePicker1.Value >
dateTimePicker2.Value && dateTimePicker1.Value > dateTimePicker3.Value &&
dateTimePicker2.Value < dateTimePicker3.Value)
            {
                tabel_vcTableAdapter.Insert(dateTimePicker1.Value,
Convert.ToInt32(textBox1.Text), dateTimePicker2.Value,
dateTimePicker3.Value);
                tabel_vcTableAdapter.Update(tabelDataSet.Tabel_vc);
                tabel_vcTableAdapter.FillBy(tabelDataSet.Tabel_vc);
            }
            else
            {
                MessageBox.Show("Помилка при введенні");
            }
            textBox1.Clear();
        }

        private void button2_Click(object sender, EventArgs e)
        {

spivrobitnukTabelTableAdapter.Insert(Convert.ToInt32(comboBox1.SelectedValu
e.ToString()), Convert.ToInt32(comboBox2.SelectedValue.ToString()));
            spivrobitnukTabelTableAdapter.Update(tabelDataSet.SpivrobitnukTabel);
            spivrobitnukTabelTableAdapter.FillBy(tabelDataSet.SpivrobitnukTabel);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (textBox2.Text.Length > 0 && (dateTimePicker2.Value <=
dateTimePicker4.Value && dateTimePicker3.Value >= dateTimePicker4.Value))
            {

vidpracTimeTableAdapter.Insert(Convert.ToInt32(comboBox3.SelectedValue.ToS
tring()), Convert.ToInt32(comboBox4.SelectedValue.ToString()),
dateTimePicker4.Value, Convert.ToInt32(textBox2.Text));
                vidpracTimeTableAdapter.Update(tabelDataSet.VidpracTime);
                vidpracTimeTableAdapter.FillBy(tabelDataSet.VidpracTime);
            }
```

```
            else
            {
                MessageBox.Show("Заповніть всі поля");
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
    public partial class Form8 : Form
    {
        public Form8()
        {
            InitializeComponent();
        }

        private void Form8_Load(object sender, EventArgs e)
        {
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"spivrobDataSet.Spivrobitnuk". При необхідності вона може бути переміщена
або видалена.
            this.spivrobitnukTableAdapter.Fill(this.spivrobDataSet.Spivrobitnuk);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.VidpracTime". При необхідності вона може бути переміщена
або видалена.
            this.vidpracTimeTableAdapter.Fill(this.tabelDataSet.VidpracTime);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути
переміщена або видалена.

this.spivrobitnukTabelTableAdapter.Fill(this.tabelDataSet.SpivrobitnukTabel);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або
видалена.
```

```
            this.tabel_vcTableAdapter.Fill(this.tabelDataSet.Tabel_vc);

            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"spivrobDataSet.Spivrobitnuk". При необхідності вона може бути переміщена
або видалена.
            this.spivrobitnukTableAdapter.FillBy(this.spivrobDataSet.Spivrobitnuk);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.VidpracTime". При необхідності вона може бути переміщена
або видалена.
            this.vidpracTimeTableAdapter.FillBy(this.tabelDataSet.VidpracTime);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.SpivrobitnukTabel". При необхідності вона може бути
переміщена або видалена.

this.spivrobitnukTabelTableAdapter.FillBy(this.tabelDataSet.SpivrobitnukTabel);
            // TODO: дана строка коду дозволяє завантажити дані до таблиці
"tabelDataSet.Tabel_vc". При необхідності вона може бути переміщена або
видалена.
            this.tabel_vcTableAdapter.FillBy(this.tabelDataSet.Tabel_vc);

            InitializeOutputData();
        }

        private void InitializeOutputData()
        {
            int SumPrus = 0;
            int SumVids = 0;
            int SumTime = 0;
            //int SumPrus1 = 0;
            //int SumVids1 = 0;
            //int SumTime1 = 0;


            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                //int count =
Int32.Parse(dataGridView1.Rows[i].Cells[5].Value.ToString());
                //SumPrus += (count);
            }

            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                //int count1 =
Int32.Parse(dataGridView1.Rows[i].Cells[6].Value.ToString());
```

```
            //SumVids += (count1);
        }

        for (int i = 0; i < dataGridView3.RowCount; i++)
        {
            int count1 =
Int32.Parse(dataGridView3.Rows[i].Cells[6].Value.ToString());
            SumTime += (count1);
        }

        label3.Text = " Відображена кількість записів : " +
dataGridView1.RowCount.ToString();
        label4.Text = " Відображена кількість записів : " +
dataGridView2.RowCount.ToString();
        label5.Text = " Відображена кількість записів : " +
dataGridView3.RowCount.ToString();

        //label4.Text = "Присутні: " + SumPrus;
        //label5.Text = "Відсутні: " + SumVids;

        label6.Text = "Відпрацьований час: " + SumTime;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        tabelvcBindingSource.Filter = "";
        InitializeOutputData();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (checkBox1.Checked)
        {
            StringBuilder sb = new StringBuilder();
            sb.AppendFormat("Date_create >= '{0}' and Date_create <= '{1}' and
Number = '{2}'", dateTimePicker1.Value, dateTimePicker2.Value,
comboBox1.SelectedValue.ToString());

            tabelvcBindingSource.Filter = sb.ToString();
            InitializeOutputData();
        }
        else if (checkBox2.Checked)
        {
            StringBuilder sb = new StringBuilder();
```

```
            sb.AppendFormat("Den_mis >= '{0}' and Den_mis <= '{1}' and
Spivrobitnuk_Name = '{2}'", dateTimePicker1.Value, dateTimePicker2.Value,
comboBox2.SelectedValue.ToString());

            fKVidpracTiIDSp0CBAE877BindingSource.Filter = sb.ToString();
            InitializeOutputData();
        }
        else
        {
            StringBuilder sb = new StringBuilder();
            sb.AppendFormat("Date_create >= '{0}' and Date_create <= '{1}'",
dateTimePicker1.Value, dateTimePicker2.Value);

            tabelvcBindingSource.Filter = sb.ToString();
            InitializeOutputData();
        }
    }

    private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        InitializeOutputData();
    }

    private void dataGridView2_CellClick(object sender,
DataGridViewCellEventArgs e)
    {
        InitializeOutputData();
    }
  }
}
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Підприємство
{
  public partial class Form9 : Form
```

```
    {
        public Form9()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters["@Param1"].Value =
dateTimePicker1.Value;
            sqlDataAdapter1.SelectCommand.Parameters["@Param2"].Value =
dateTimePicker2.Value;
            dataSet51.Clear();
            sqlDataAdapter1.Fill(dataSet51.Mark);

            int[] gridI = new int[dataGridView1.RowCount];
            string[] gridS = new string[dataGridView1.RowCount];
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                gridS[i] =
Convert.ToString(dataGridView1.Rows[i].Cells["Expr2"].Value);
                gridI[i] =
Convert.ToInt32(dataGridView1.Rows[i].Cells["Expr1"].Value);}
            chart1.Series[0].Points.DataBindXY(gridS, gridI);}

        private void Form9_Load(object sender, EventArgs e)
        {
            sqlDataAdapter1.SelectCommand.Parameters["@Param1"].Value =
dateTimePicker1.Value;
            sqlDataAdapter1.SelectCommand.Parameters["@Param2"].Value =
dateTimePicker2.Value;
            sqlDataAdapter1.Fill(dataSet51.Mark);}
    }
}
```